WEEK 4

Model Development

Learning Objectives

In this module you will learn about:

1. Simple and Multiple Linear Regression
2. Model Evaluation using Visualization
3. Polynomial Regression and Pipelines
4. R-squared and MSE for In-Sample Evaluation
5. Prediction and Decision Making

- Question
  - **How can you determine a fair value for a used car?**

Model Development

- A model can be thought of as a mathematical equation used to predict a value given one or more other values
- Relating one or more independent variables to dependent variables.

independent variables or features

`'highway-mpg'`

$55\ mpg$

Model

dependent variables

`'predicted price'`

$5000

Model Development

- Usually the more **relevant data** you have the more accurate your model is

'highway-mpg'
'curb-weight'
'engine-size'
'highway-mpg'

Model

'price'

$5400

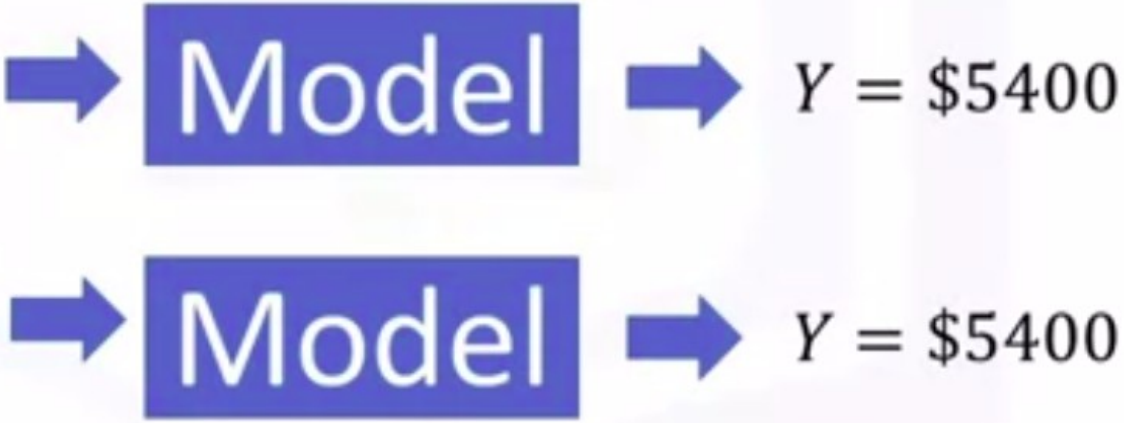
Model Development

To understand why more data is important consider the following situation:

1. you have two almost identical cars
2. Pink cars sell for significantly less

'highway-mpg'
'curb-weight'
'engine-size'
'highway-mpg'

'highway-mpg'
'curb-weight'
'engine-size'
'highway-mpg'



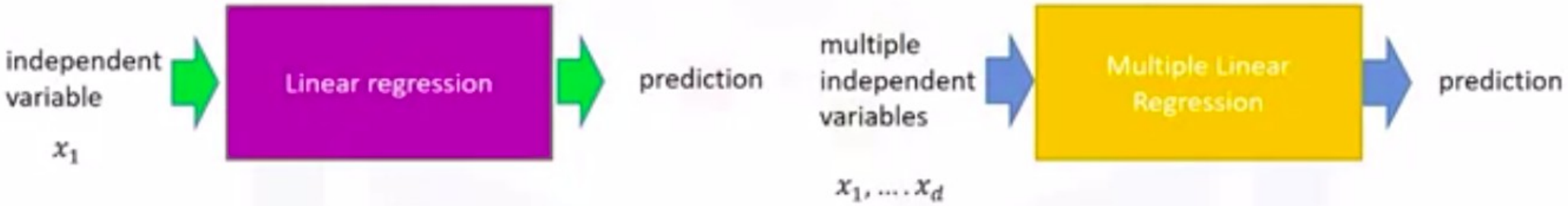
$Y = \$5400$

$Y = \$5400$

Linear Regression and Multiple Linear Regression

Introduction

- Linear regression will refer to one independent variable to make a prediction
- Multiple Linear Regression will refer to multiple independent variables to make a prediction



Sir

1.
2.

- b_0
- b_1

Question

what kind of relationship would we like between the predictor variable *x* and the dependent variable *y*

- (●) linear
- ( ) Non-linear
- ( ) logarithmic

Correct
correct

Skip Continue

CLASS.ai

Simple Linear Regression

1. The predictor (independent) variable - x
2. The target (dependent) variable - y

$$y = b_0 + b_1 x$$

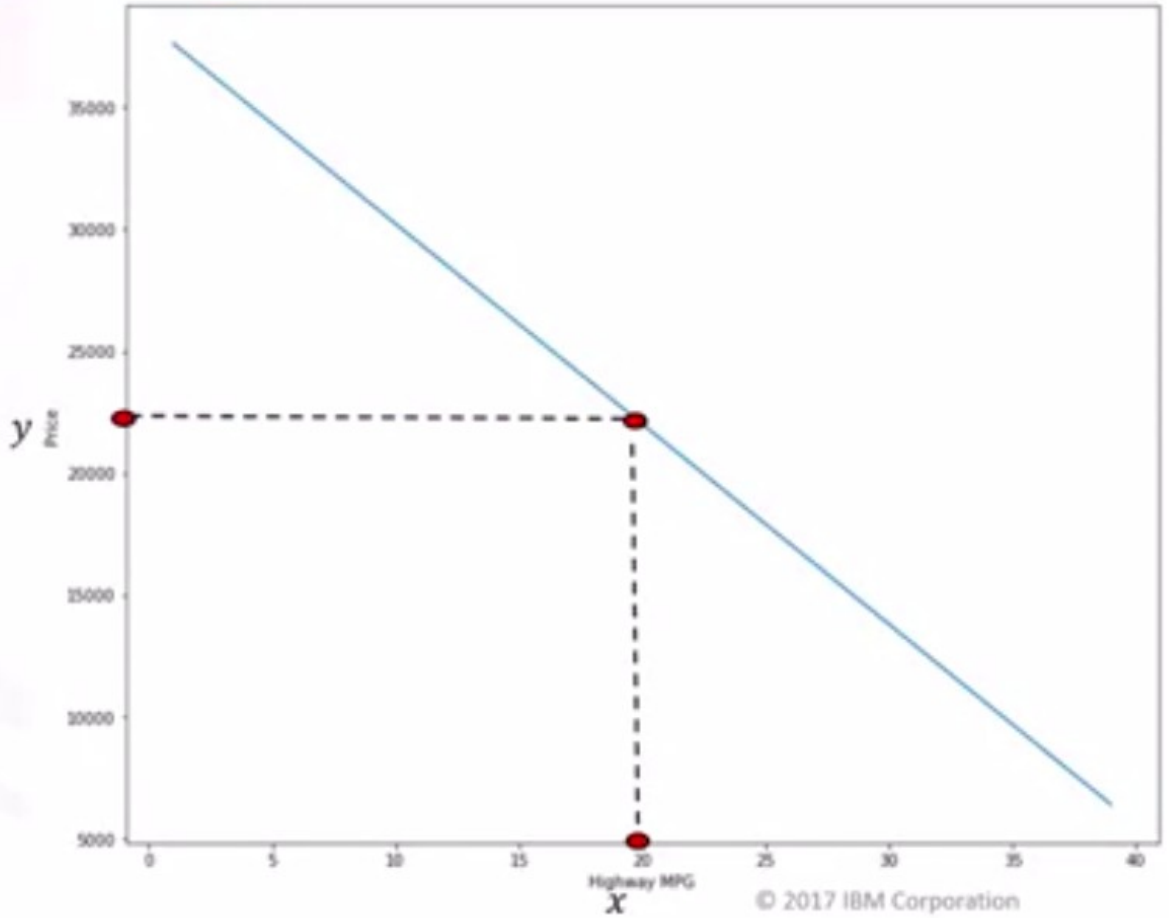
- b_0: **the intercept**
- b_1 : **the slope**

Simple Linear Regression: Prediction

$$y = 38423 - 821x$$

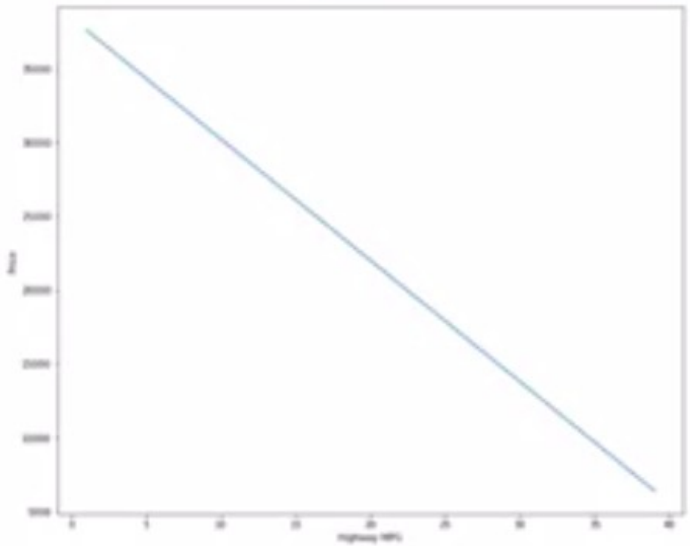
$$= 38423 - 821\ (20)$$

$$= 22\ 003$$



COGNITIVE CLASS.ai

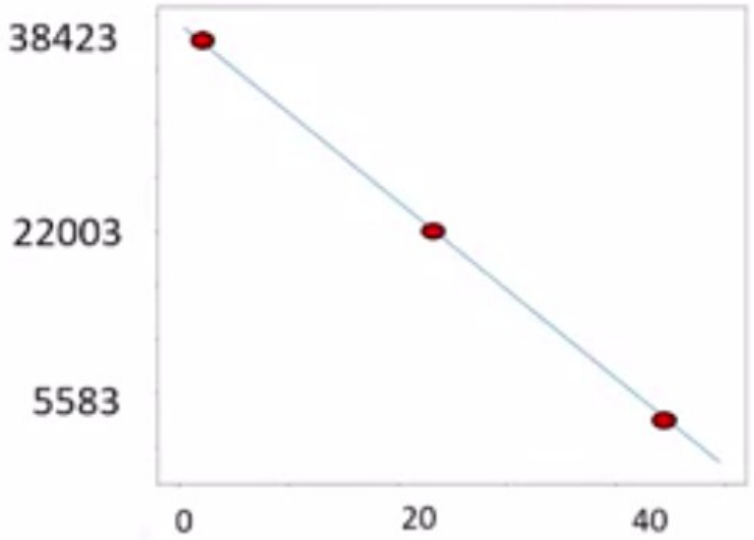
Simple Linear Regression: Fit



Fit

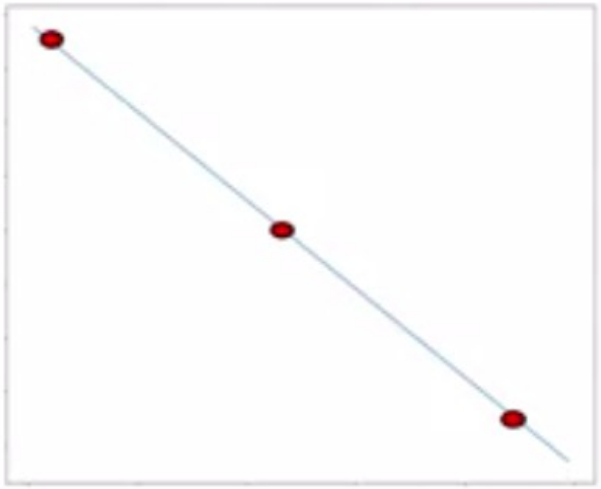
(b_0, b_1)

Simple Linear Regression: Fit



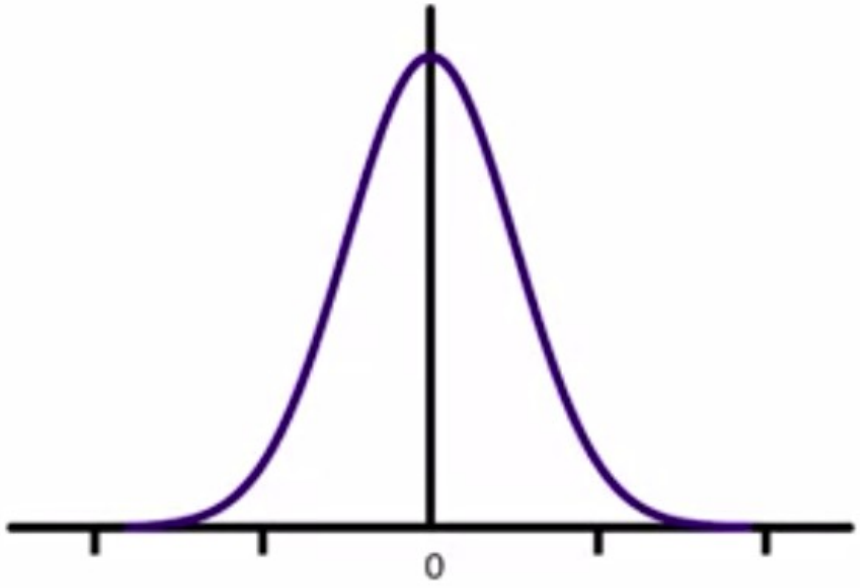
$$X = \begin{bmatrix} \\ \\ \end{bmatrix} \quad Y = \begin{bmatrix} \\ \\ \end{bmatrix}$$

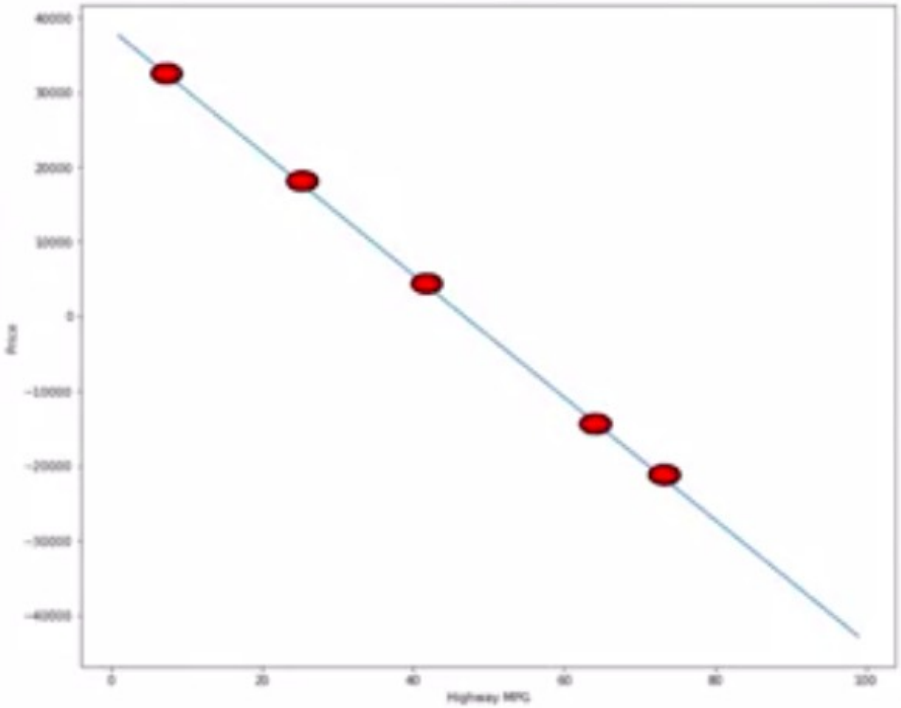
Simple Linear Regression: Fit



$$X = \begin{bmatrix} 0 \\ 20 \\ 40 \end{bmatrix} \quad Y = \begin{bmatrix} 38423 \\ 22003 \\ 5583 \end{bmatrix}$$

Simple Linear Regression: Fit

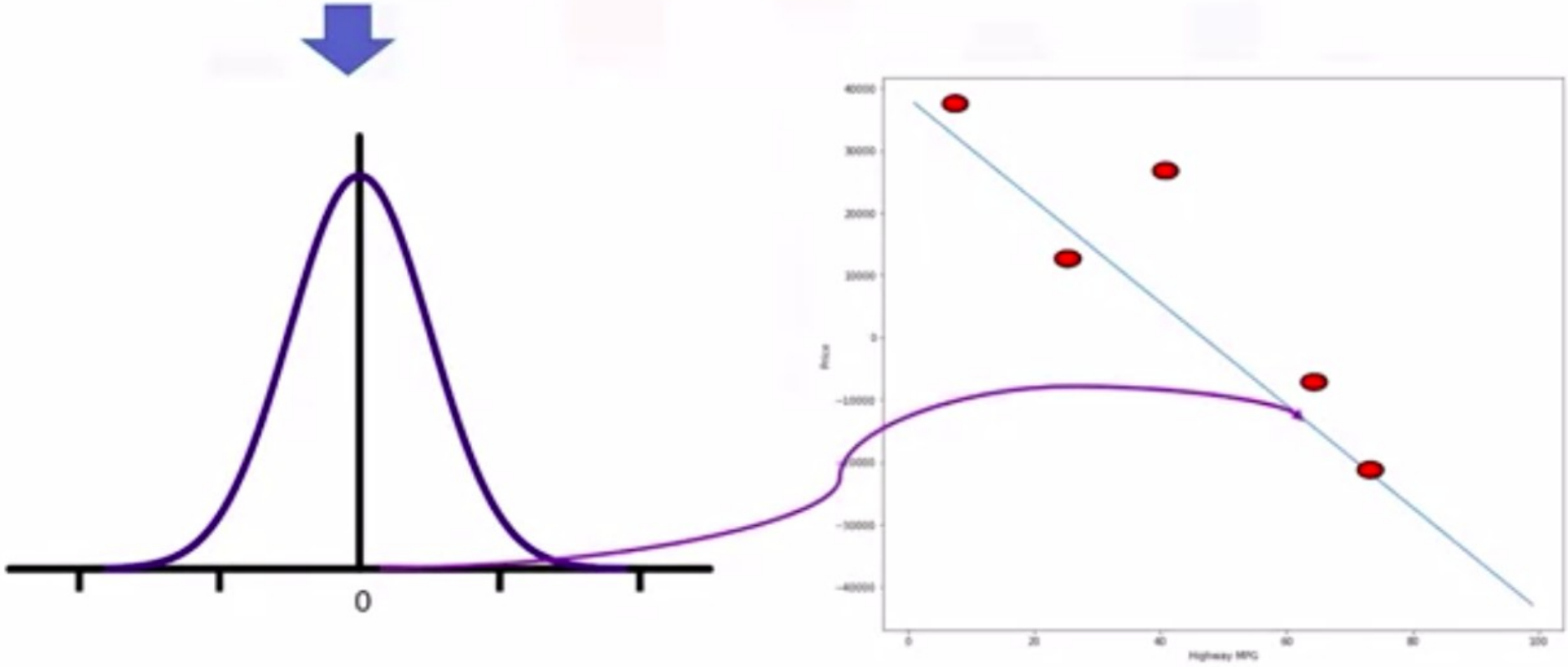




COGNITIVE CLASS.ai

x

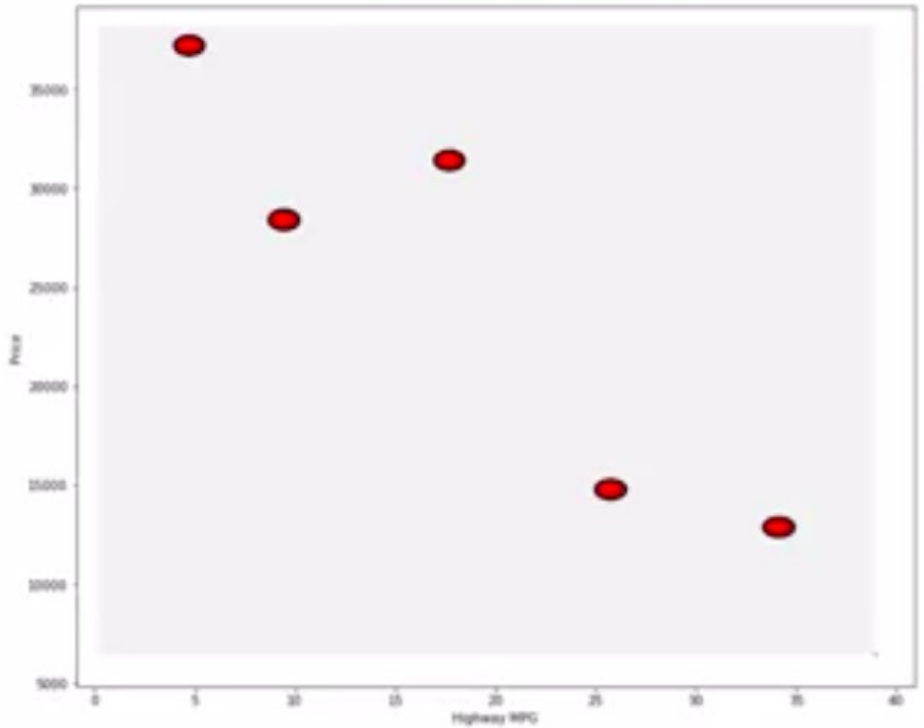
Simple Linear Regression: Fit



COGNITIVE CLASS.ai

x

Simple Linear Regression

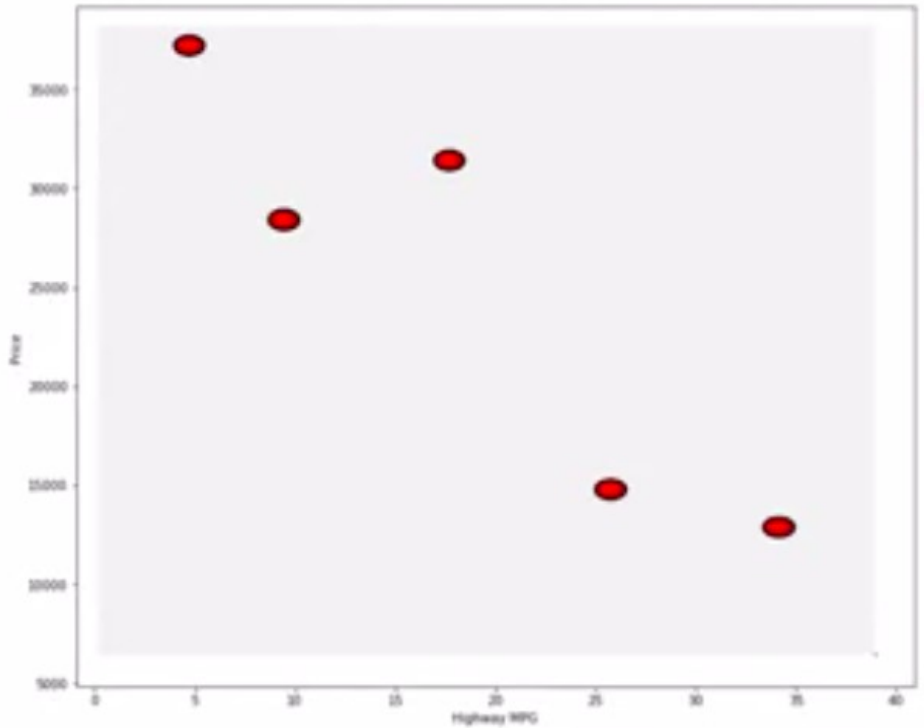


Fit

Predict

COGNITIVE CLASS.ai

Simple Linear Regression

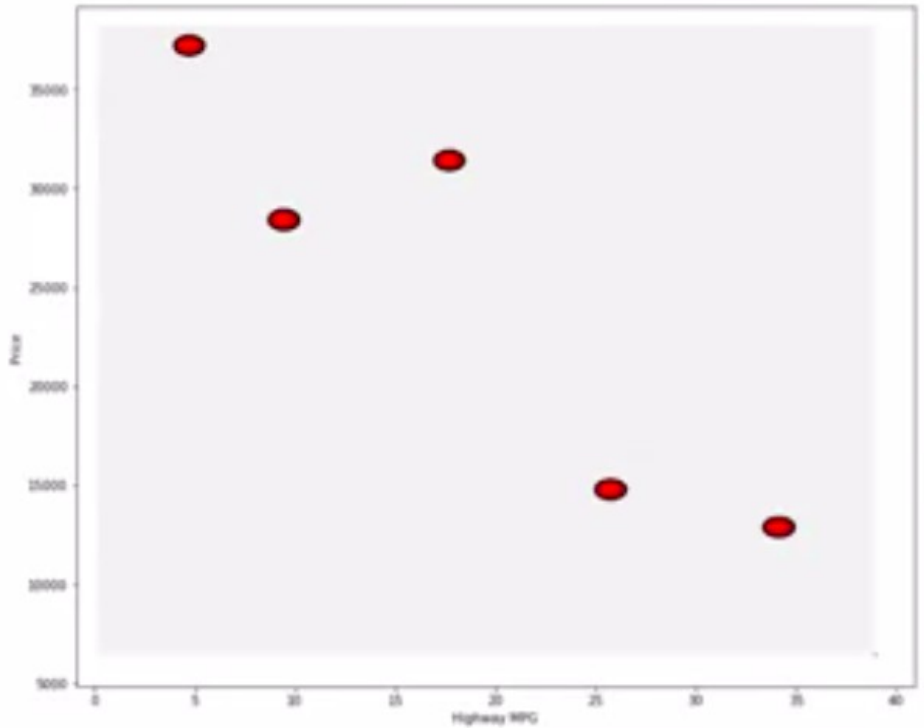


Fit

(b_0, b_1)

Predict

Simple Linear Regression

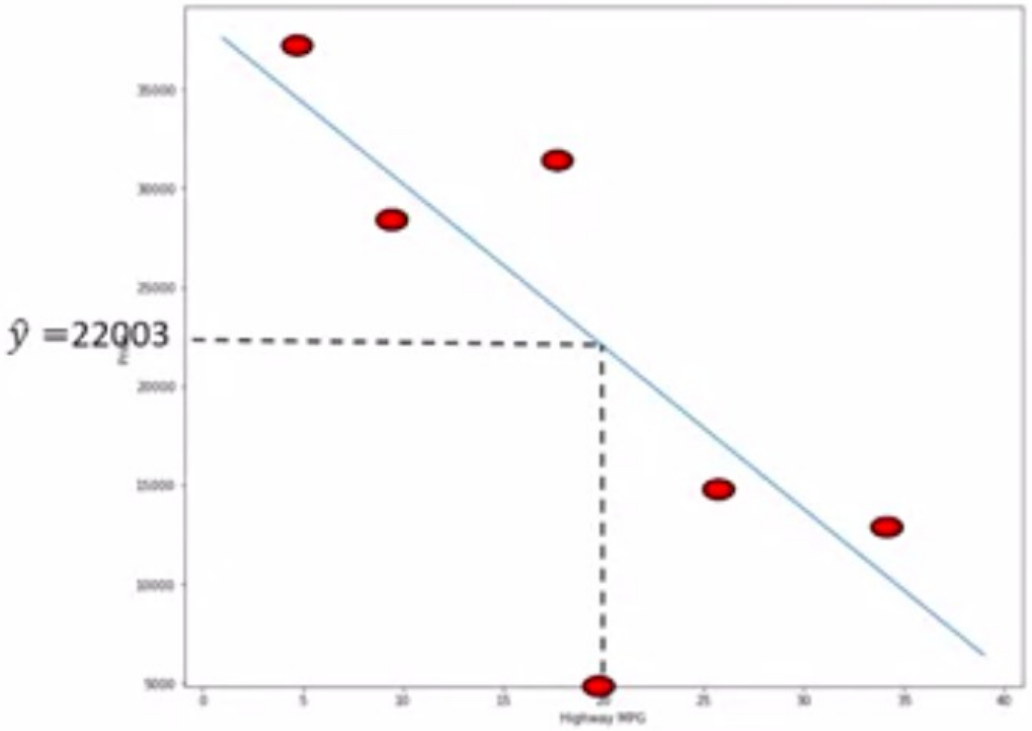


Fit

Predict

$$\hat{y} = b_0 + b_1 x$$

Simple Linear Regression

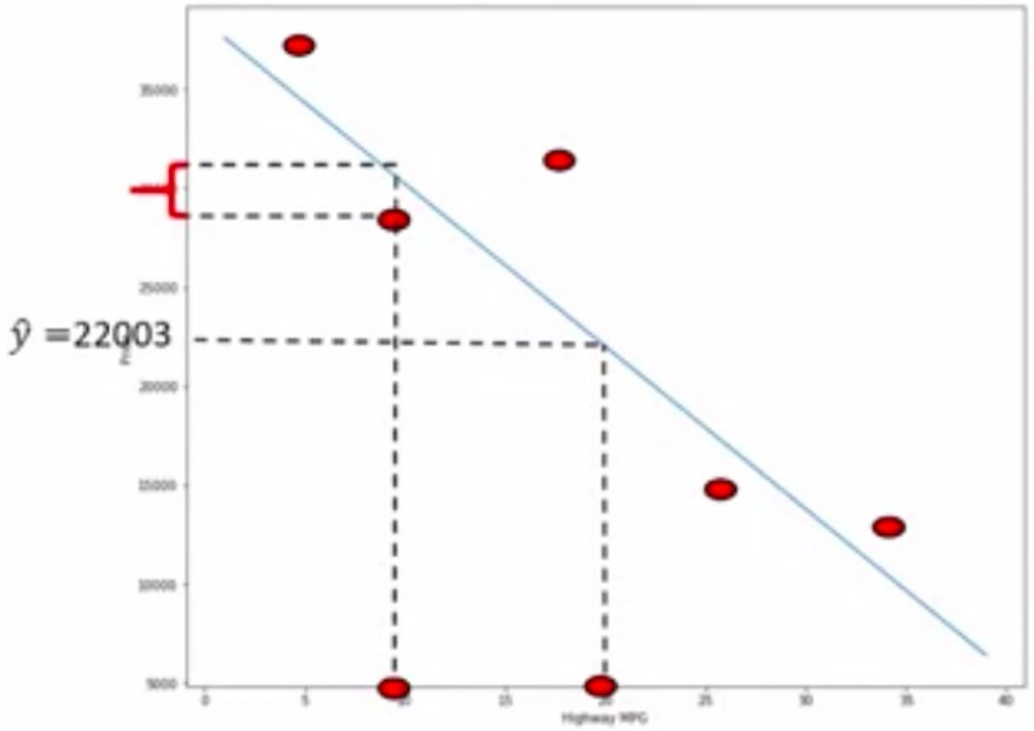


Fit

Predict

$$\hat{y} = b_0 + b_1 x$$

Simple Linear Regression



Fit

Predict

$$\hat{y} = b_0 + b_1 x$$

Fitting a Simple Linear Model Estimator

- X :Predictor variable
- Y: Target variable

1. Import linear_model from scikit-learn

```
from sklearn.linear_model import LinearRegression
```

2. Create a Linear Regression Object using the constructor :

```
lm=LinearRegression()
```

Question

Consider the following lines
of code

```
1  from sklearn.linear_model import LinearRegression
2
3  lm=LinearRegression()
4  X = df[['highway-mpg']]
5  Y = df['price']
6
```

Type the next line of code to fit the model

lm.fit(x,y)

Correct

Correct you just apply the method fit with the variables X
and Y as arguments

Skip Continue

Fitting a Simple Linear Model

- We define the predictor variable and target variable

```
X = df[['highway-mpg']]
Y = df['price']
```

- Then use lm.fit (X, Y) to fit the model , i.e fine the parameters b_0 and b_1

```
lm.fit(X, Y)
```

- We can obtain a prediction

```
Yhat=lm.predict(X)
```

Yhat	X
2	5
:	
3	4

SLR – Estimated Linear Model

- We can view the intercept (b_0):

```
lm.intercept_
38423.305858
```

- We can also view the slope (b_1):

```
lm.coef_
-821.73337832
```

- The Relationship between Price and Highway MPG is given by:
- **Price = 38423.31 - 821.73 * highway-mpg**

$$\hat{Y} = b_0 + b_1 x$$

Multiple Linear Regression (MLR)

This method is used to explain the relationship between:

- One continuous target (Y) variable
- Two or more predictor (X) variables

Multiple Linear Regression (MLR)

$$\hat{Y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

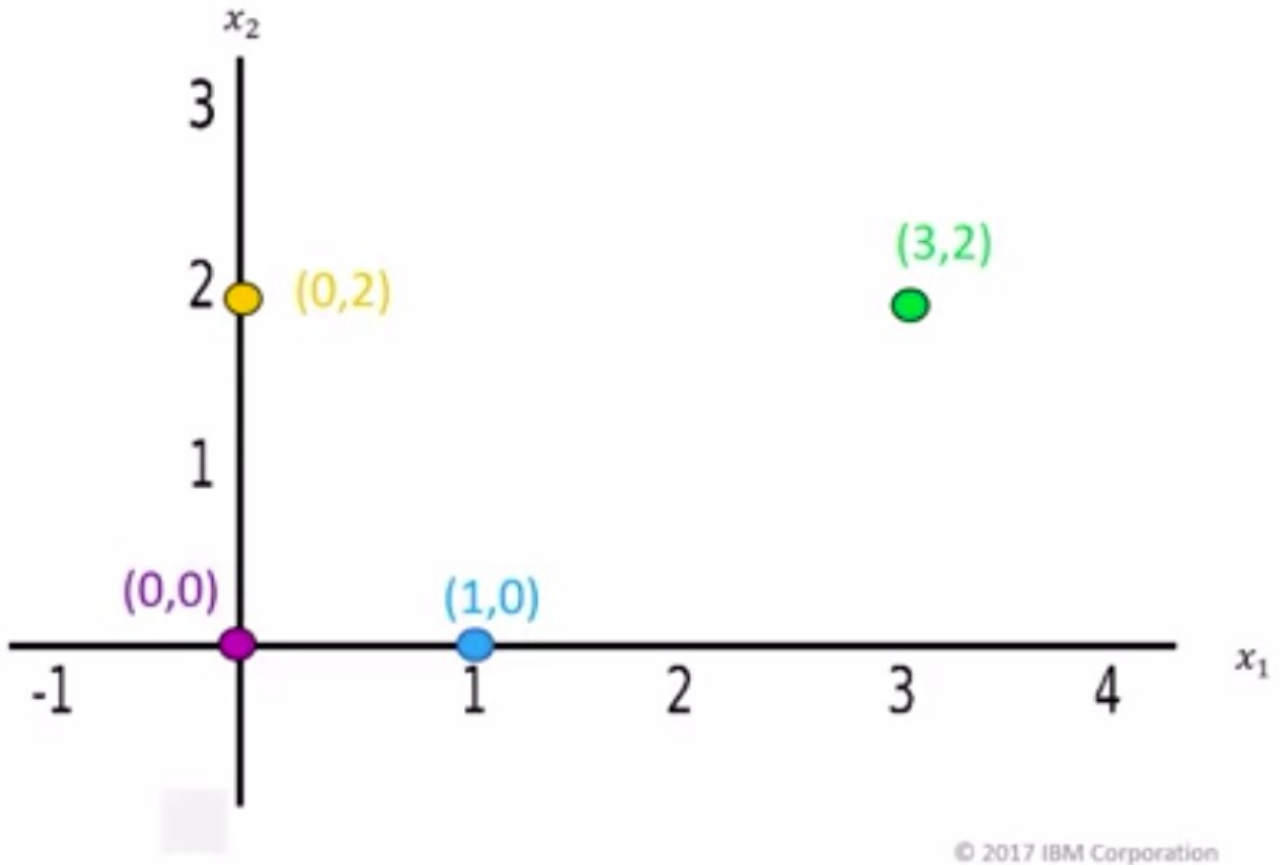
- b_0 : **intercept (X=0)**
- b_1: the **coefficient** or **parameter** of x_1
- b_2: the **coefficient** of **parameter** x_2 and so on..

Multiple Linear Regression (MLR)

$$\hat{Y} = 1 + 2x_1 + 3x_2$$

- The variables x_1 and x_2 can be visualized on a 2D plane, lets do an example on the next slide

n	x_1	x_2
1	0	0
2	0	2
3	1	0
4	3	2



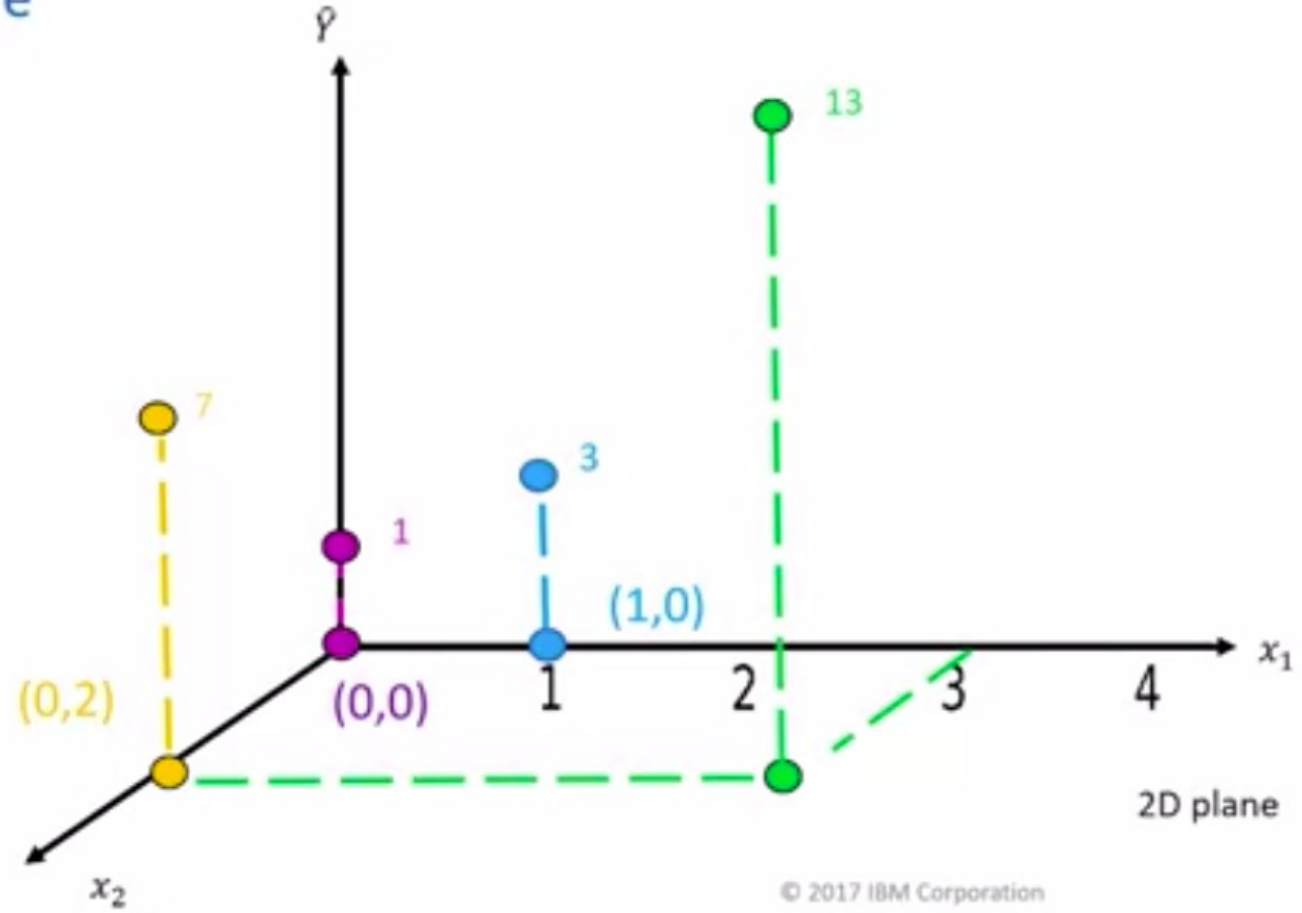
COGNITIVE CLASS.ai

- This is shown below where

$$\hat{Y} = 1 + 2x_1 + 3x_2$$

n	x_1	x_2
1	0	0
2	0	2
3	1	0
4	3	2

$\hat{Y}$
1
7
3
13



Fitting a Multiple Linear Model Estimator

1. We can extract the for 4 predictor variables and store them in the variable Z

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

2. Then train the model as before:

```
lm.fit(Z, df['price'])
```

3. We can also obtain a prediction

```
Yhat=lm.predict(X)
```

x_1	x_2	x_3	x_4
3	5	-4	3
:	:	:	:
2	4	2	-4

Yhat
2
:
3

MLR – Estimated Linear Model

1. Find the intercept (b_0)

```
lm.intercept_
-15678.742628061467
```

2. Find the coefficients (b_1, b_2, b_3, b_4)

```
lm.coef_
array([52.65851272 ,4.69878948,81.95906216 , 33.58258185])
```

The Estimated Linear Model:

- **Price** = -15678.74 + (52.66) * **horsepower** + (4.70) * **curb-weight** + (81.96) * **engine-size** + (33.58) * **highway-mpg**

$$\hat{Y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

Back

Practice Quiz: Linear Regression and Multiple Linear Regression

Practice Quiz • 6 min • 2 total points

Congratulations! You passed!

Grade received 100% **To pass** 50% or higher

Go to next item

1. consider the following lines of code, what is the name of the column that contains the target values:

1 / 1 point

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
X = df[['highway-mpg']]
Y = df['price']
lm.fit(X, Y)
Yhat=lm.predict(X)
```

- (●) 'price'
- ( ) 'highway-mpg'

Correct

correct

Back

Practice Quiz: Linear Regression and Multiple Linear Regression

Practice Quiz • 6 min • 2 total points

highway mpg

Correct
correct

2. consider the following equation: **1 / 1 point**

$$y = b_0 + b_1 x$$

the variable y is?

- the predictor or independent variable
- the intercept
- the target or dependent variable

Correct
correct

Model Evaluation Using Visualization

IBM Developer

SKILLS NETWORK

Regression Plot

Why use regression plot?

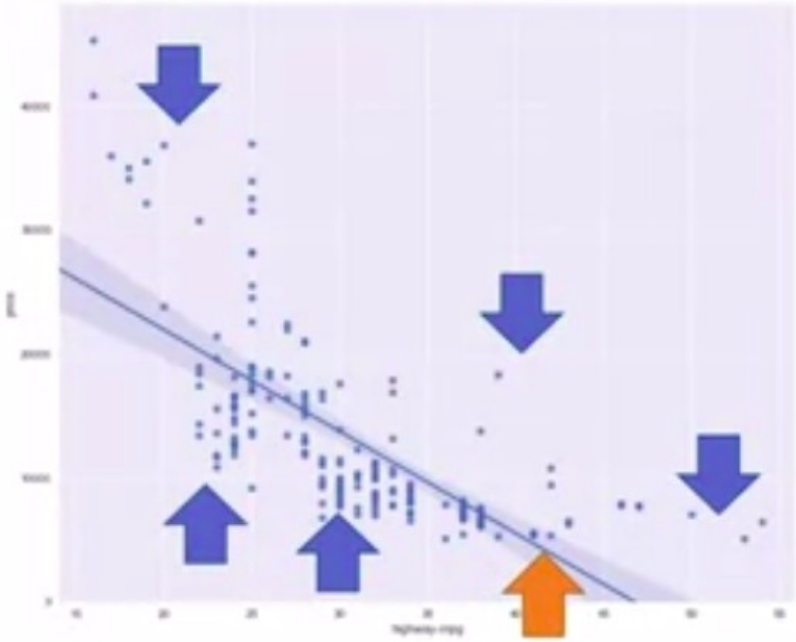
It gives us a good estimate of:

1. The relationship between two variables
2. The strength of the correlation
3. The direction of the relationship (positive or negative)

Regression Plot

Regression Plot shows us a combination of:

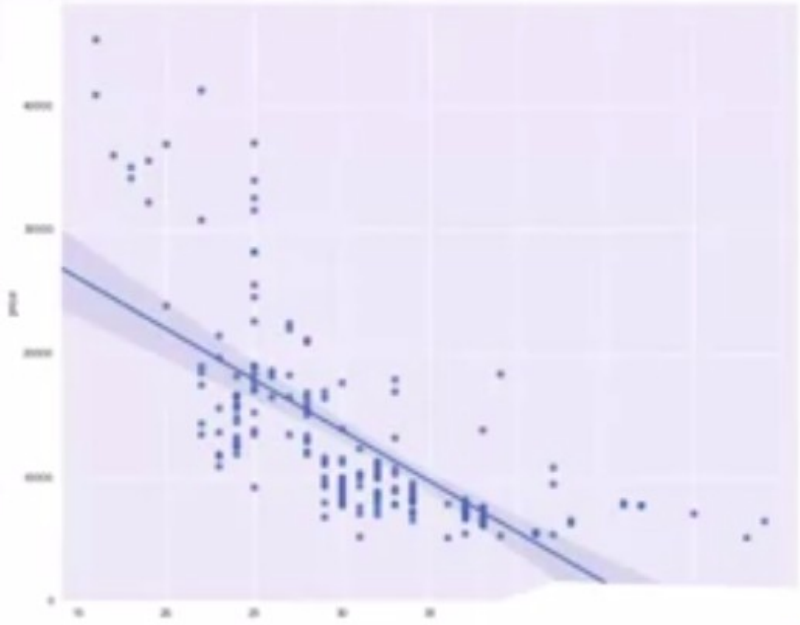
- The scatterplot: where each point represents a different y
- The fitted linear regression line ($\hat{y}$).



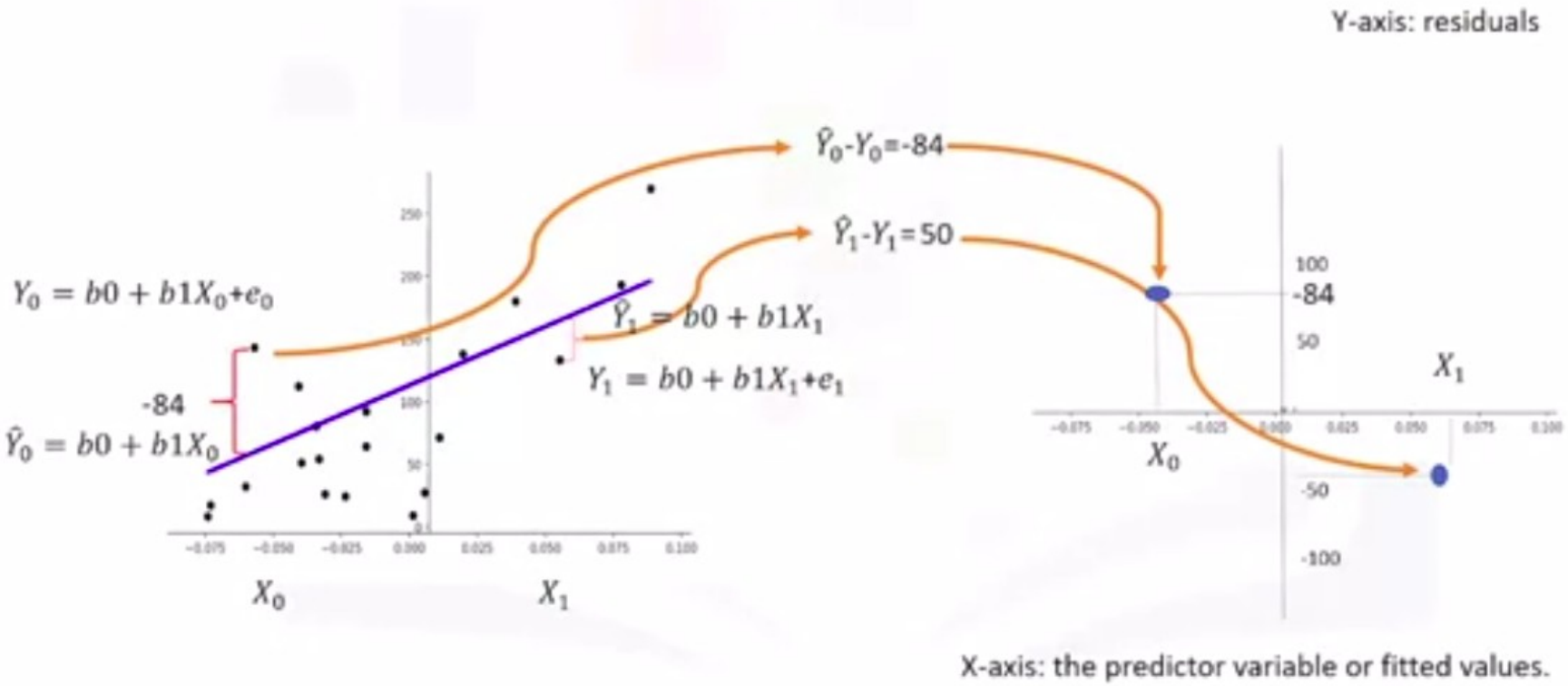
Regression Plot

```
import seaborn as sns
```

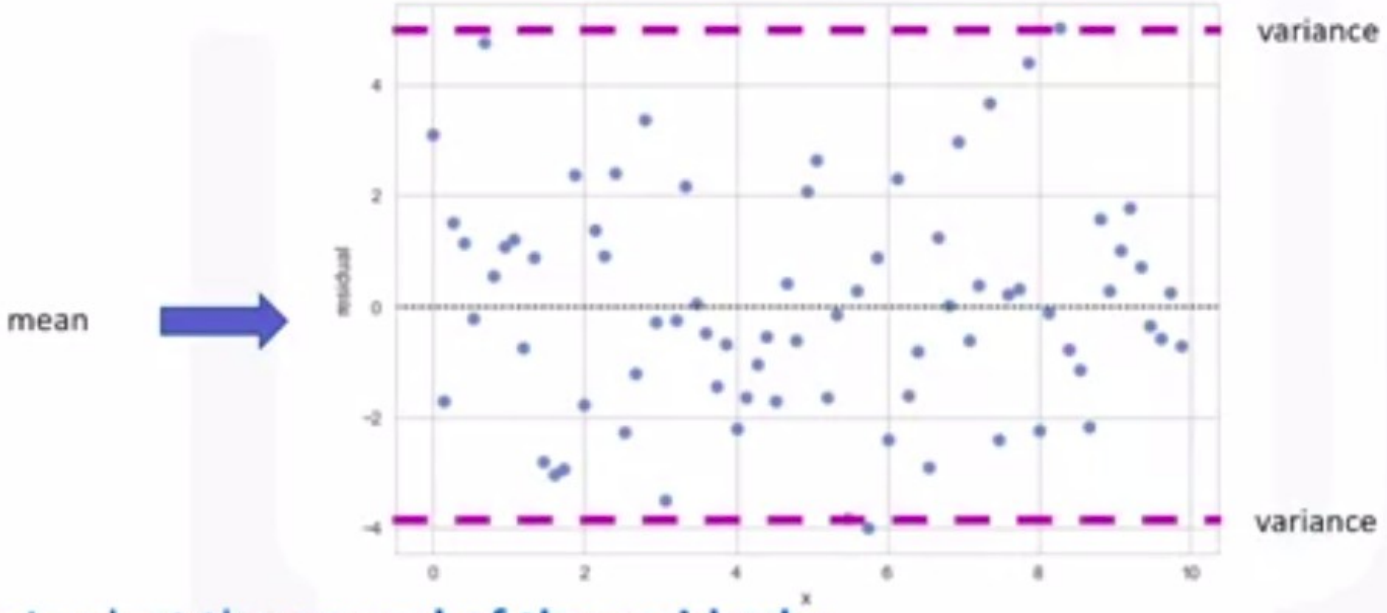
```
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```



Residual Plot

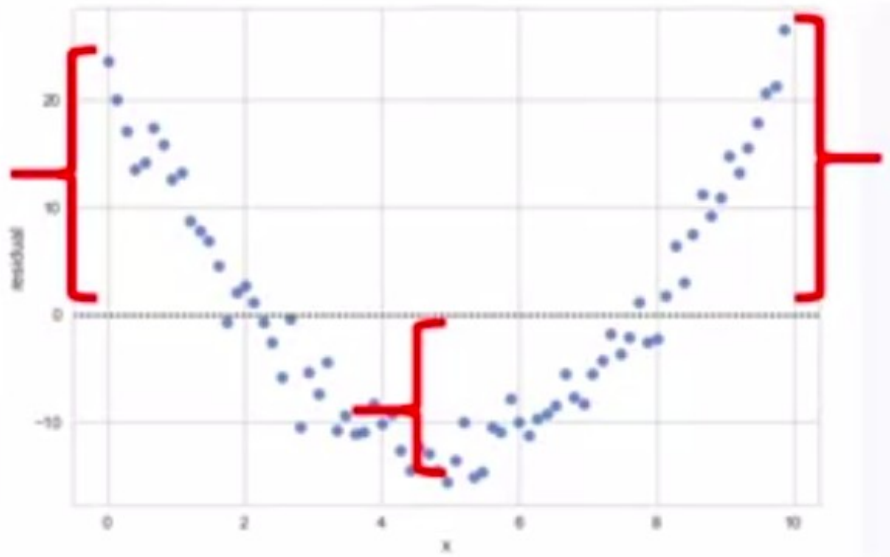


Residual Plot



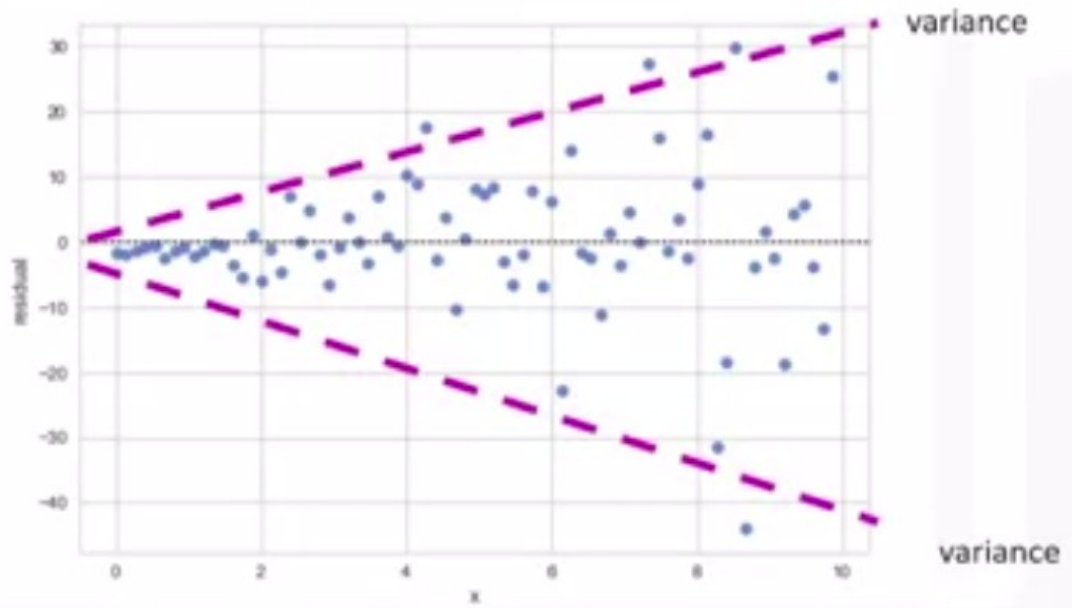
- Look at the **spread of the residuals**:
  - Randomly spread out around x-axis then a linear model is appropriate.

Residual Plot



- Not randomly spread out around the x-axis
- Nonlinear model may be more appropriate

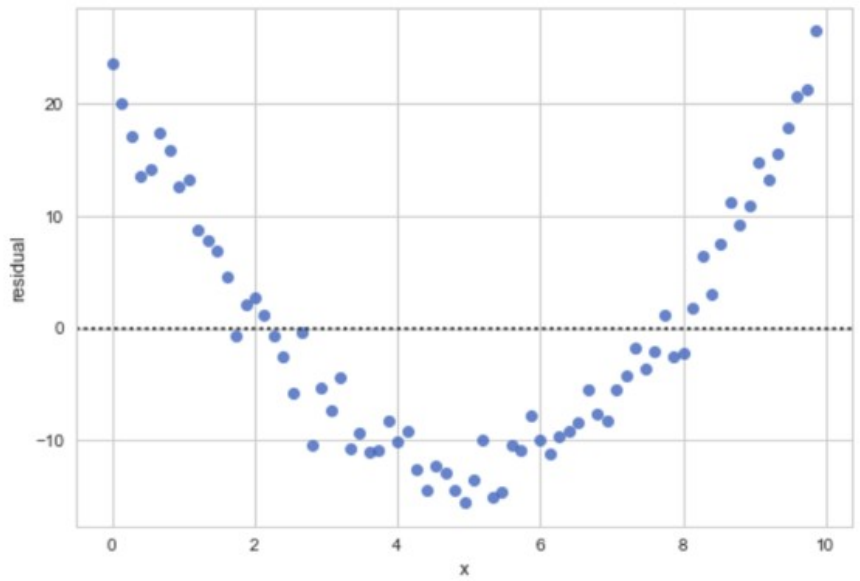
Residual Plot



- Not randomly spread out around the x-axis
- Variance appears to change with x axis

Question

consider the following residuals plot, is the linear assumption correct:



- ○ yes
- ◉ no

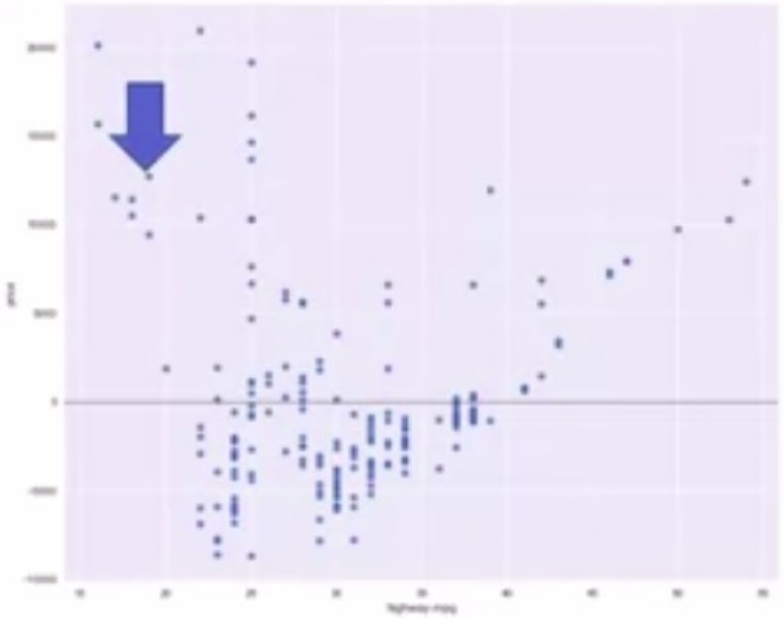
Correct
correct

Skip Continue

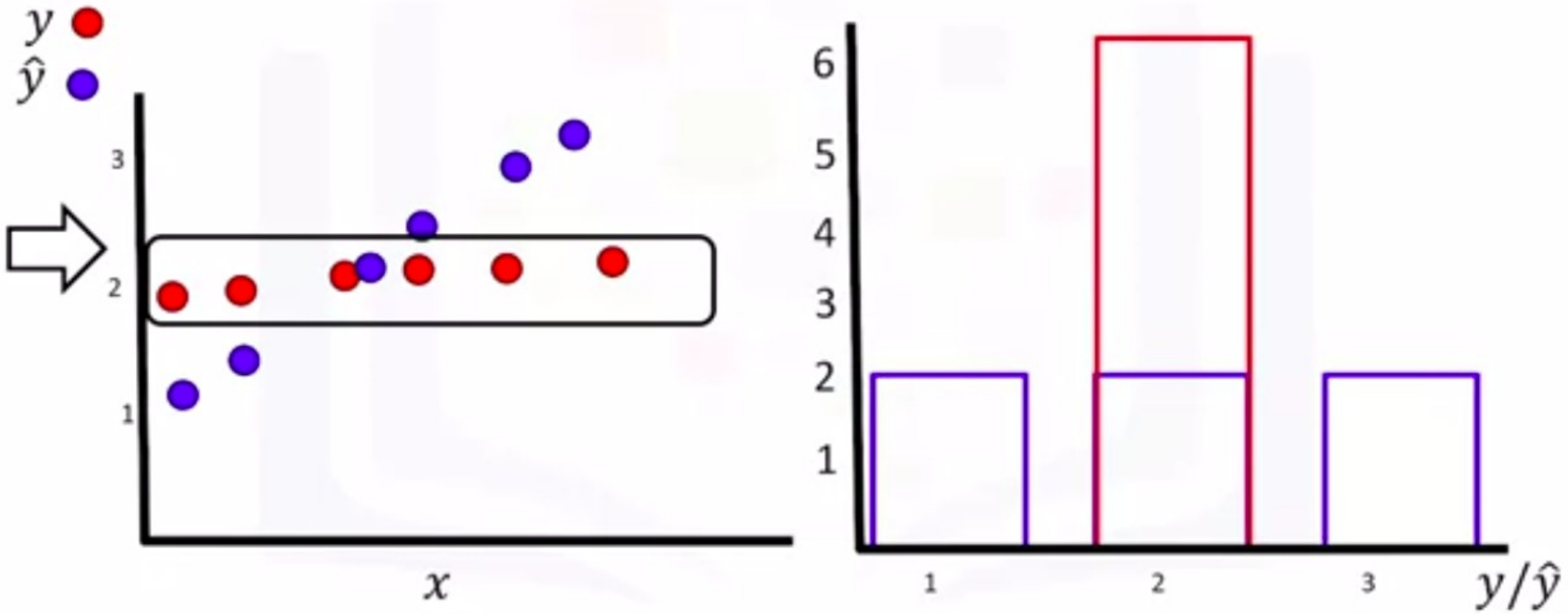
Residual Plot

```
import seaborn as sns
```

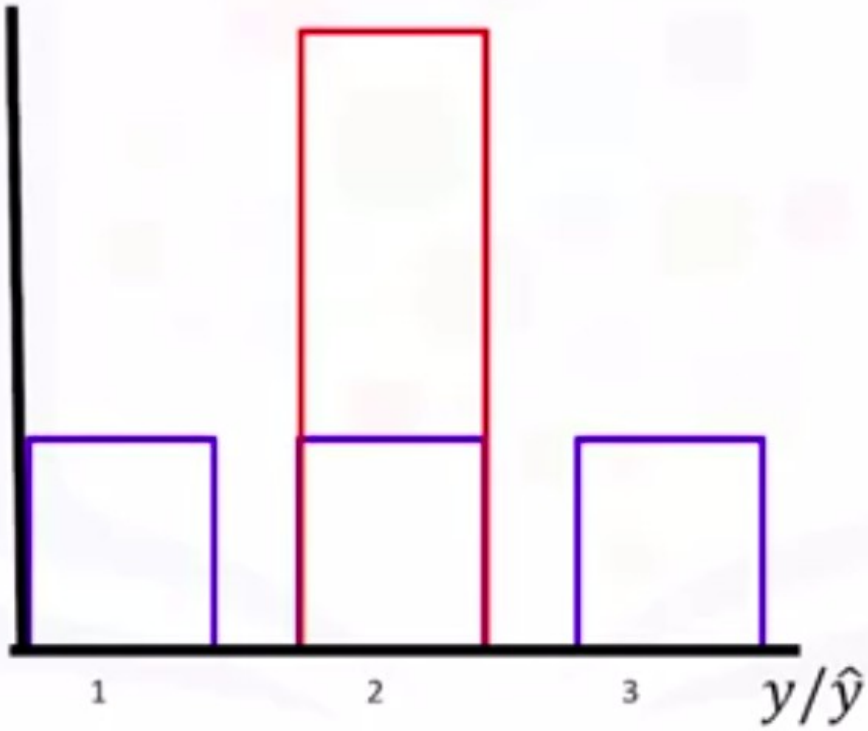
```
sns.residplot(df['highway-mpg'], df['price'])
```



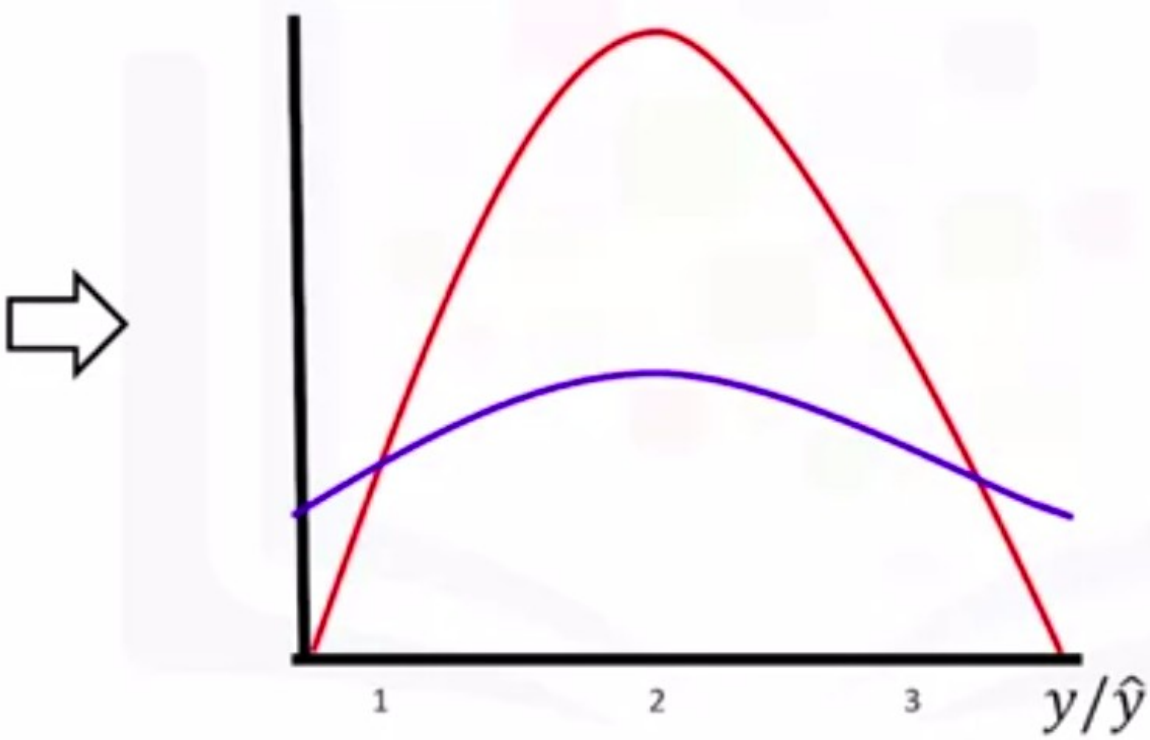
Distribution Plots



Distribution Plots



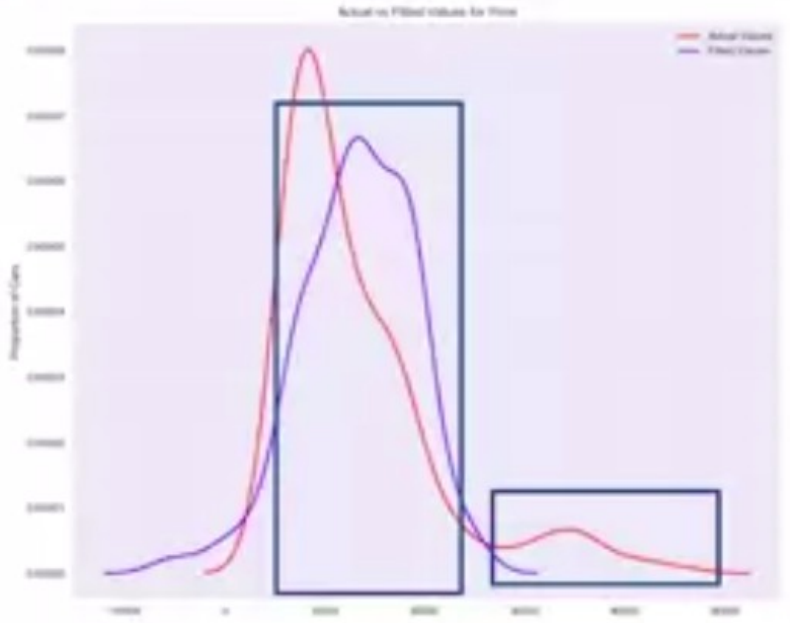
Distribution Plots



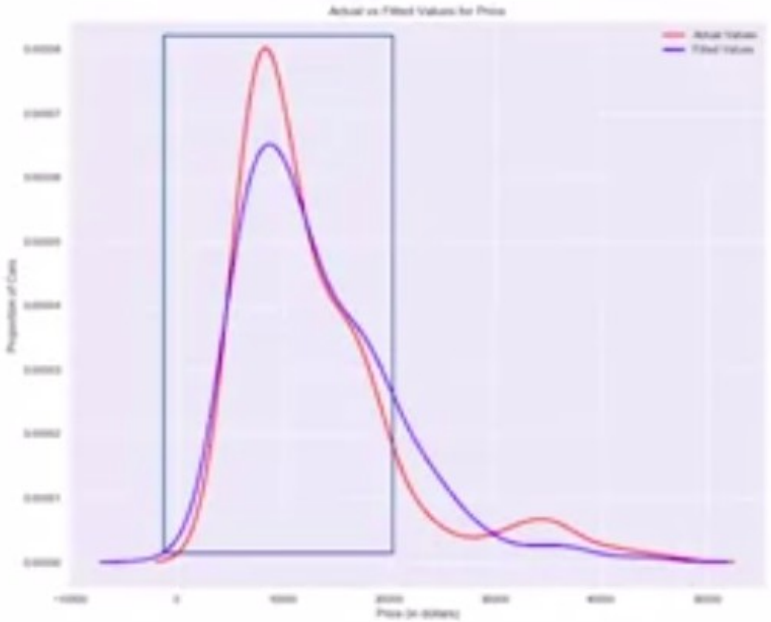
Distribution Plots

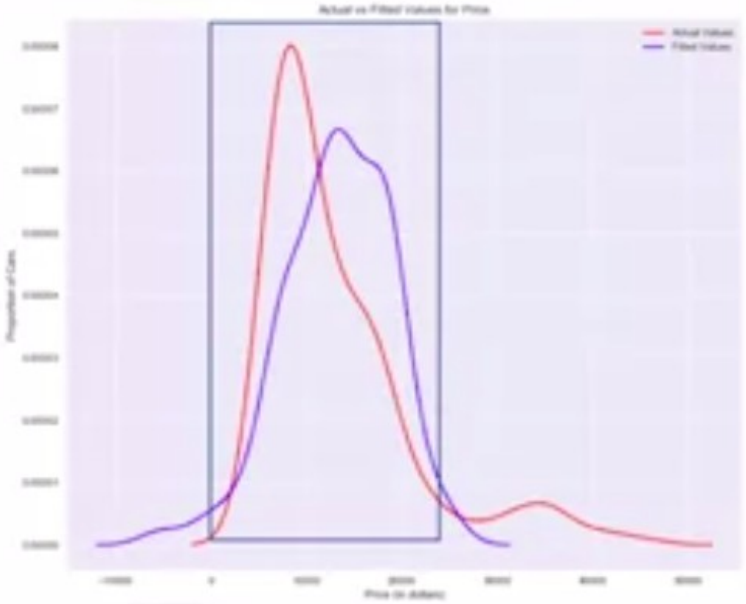
Compare the distribution plots:

- The fitted values that result from the model
- The actual values



MLR – Distribution Plots





Distribution Plots

```
import seaborn as sns
```

```
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")

sns.distplot(Yhat, hist=False, color="b", label="Fitted Values" , ax=ax1)
```

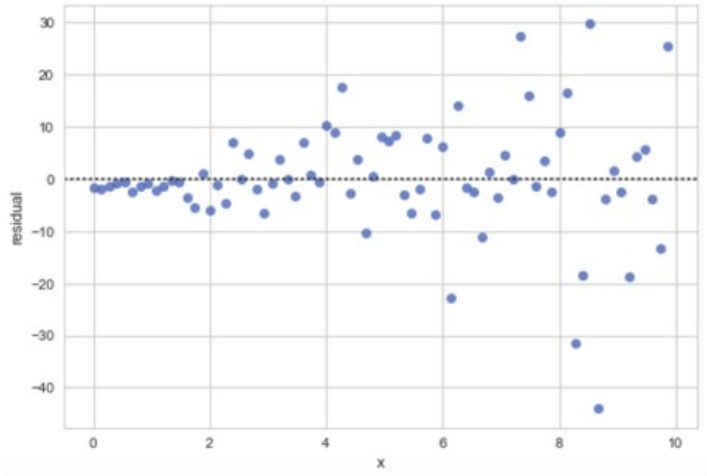
Back

Practice Quiz: Model Evaluation using Visualization

Practice Quiz • 3 min • 1 total point

1. Consider the following **Residual Plot**, is our linear model correct :

1 / 1 point



- (●) no
- ( ) yes

Correct

correct

Polynomial Regression and Pipelines

IBM Developer

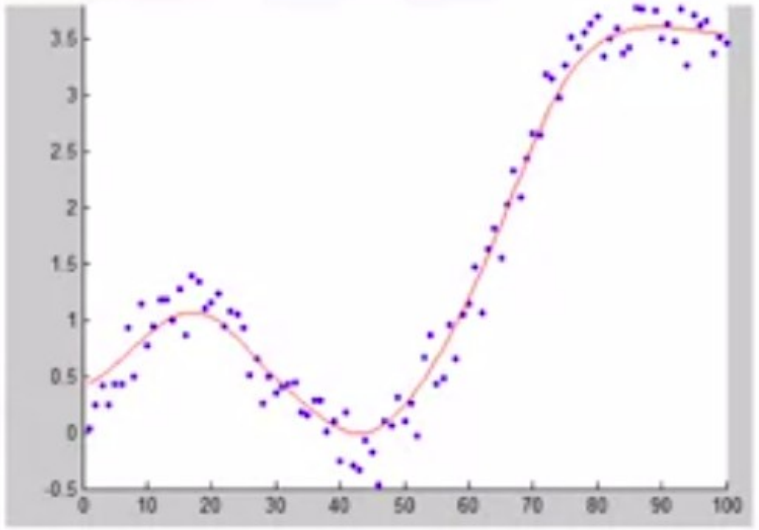
SKILLS NETWORK

Polynomial Regressions

- A special case of the general linear regression model
- Useful for describing curvilinear relationships

Curvilinear relationships:

By squaring or setting higher-order terms of the predictor variables



Polynomial Regression

- Quadratic – 2nd order

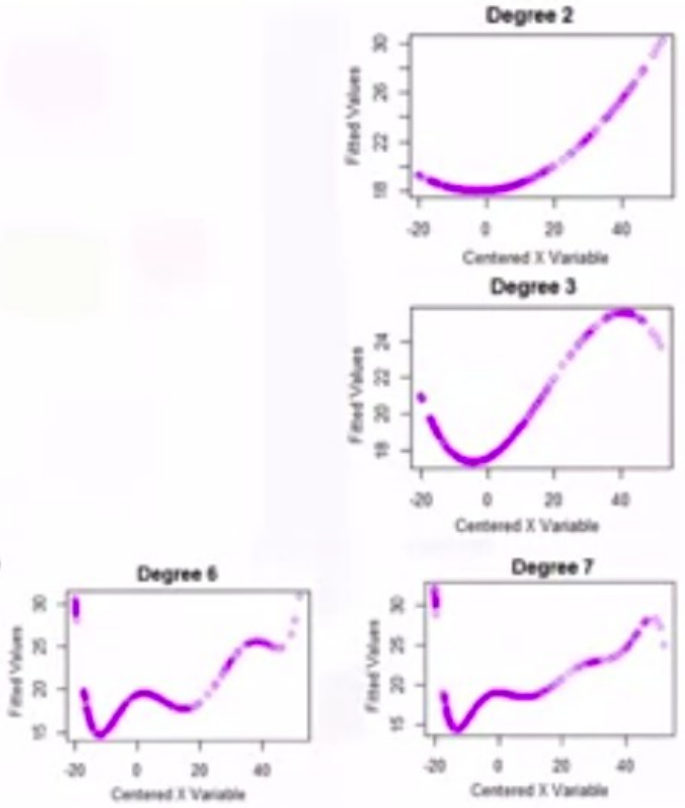
$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2$$

- Cubic – 3rd order

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3$$

- Higher order

$$\hat{Y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3 + ..$$



Polynomial Regression

1. Calculate Polynomial of 3rd order

```
f=np.polyfit(x,y,3)

p=np.poly1d(f)
```

2. We can print out the model

```
print (p)
```

$$-1.557(x_1)^3 + 204.8(x_1)^2 + 8965\,x_1 + 1.37\text{x}10^5$$

Polynomial Regression

- We can also have multi dimensional polynomial linear regression

Polynomial Regression with More than One Dimension

- We can also have multi dimensional polynomial linear regression

$$\hat{Y} = b0 + b1\,X_1 + b2\,X_2 + b3\,X_1\,X_2 + b4(X_1)^2 + b5(X_2)^2 + ..$$

Polynomial Regression with More than One Dimension

- The "preprocessing" library in scikit-learn,

```
from sklearn.preprocessing import PolynomialFeatures

pr=PolynomialFeatures(degree=2, include_bias=False)

x_polly=pr.fit_transform(x[['horsepower', 'curb-weight']])
```

Polynomial Regression with More than One Dimension

```
pr=PolynomialFeatures(degree=2)
```

X_1	X_2
1	2

Question

How would you create a second order polynomial transform object **pr**:

```
1  pr=PolynomialFeatures(degree=2)
```

```
1  xhat=pr.fit_transform([1,2], include_bias=False)
2
```

Correct

correct

Skip Continue

Polynomial Regression with More than One Dimension

```
pr=PolynomialFeatures(degree=2)
```

X_1	X_2
1	2

```
pr=PolynomialFeatures(degree=2,include_bias=False)
pr.fit_transform([[1,2]])
```

X_1	X_2	X_1X_2	X_1^2	X_2^2
1	2	(1) 2	1	$(2)^2$

1	2	2	1	4

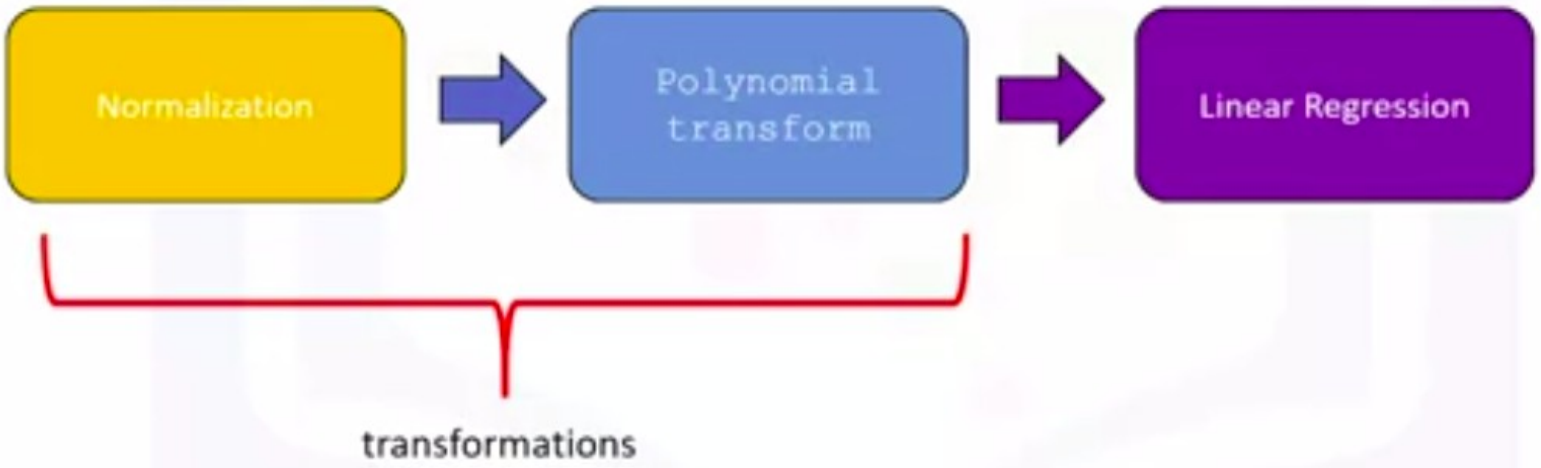
Pre-processing

- For example we can Normalize the each feature simultaneously

```
from sklearn.preprocessing import StandardScaler
SCALE=StandardScaler()
SCALE.fit(x_data[['horsepower', 'highway-mpg']])
x_scale=SCALE.transform(x_data[['horsepower', 'highway-mpg']])
```

Pipelines

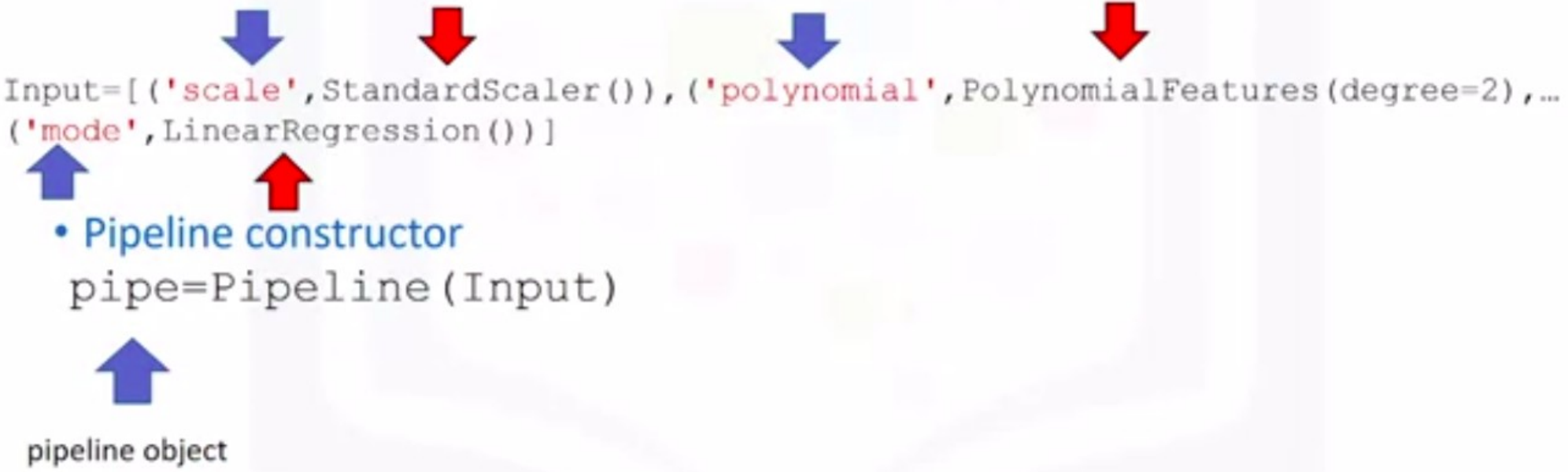
- There are many steps to getting a prediction



Pipelines

```
from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import Pipeline
```

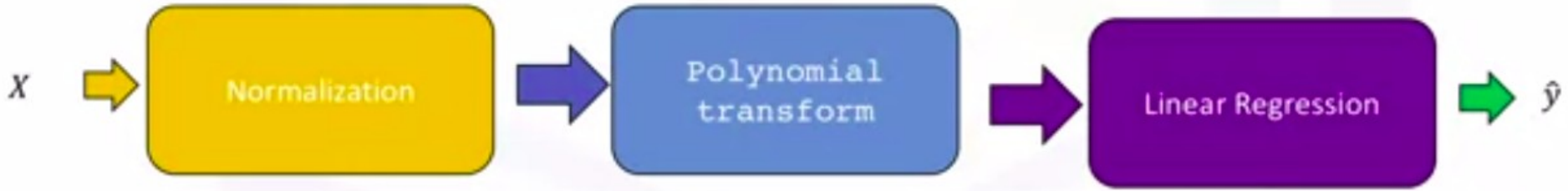
Pipeline Constructor



Pipeline Constructor

- We can train the pipeline object

```
Pipe.fit(df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y)

yhat=Pipe.predict(X[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
```



Back

Practice Quiz: Polynomial Regression and Pipelines

Practice Quiz • 3 min • 1 total point

Congratulations! You passed!

Grade received 100% **To pass** 50% or higher

Go to next item

1. What functions are used to generate Polynomial Regression with more than one dimension

1 / 1 point

○
```
f=np.polyfit(x,y,3)
p=np.poly1d(f)

```

◉
```
pr=PolynomialFeatures(degree=2)
 pr.fit_transform([1,2], include_bias=False)
```

Measures for In-Sample Evaluation

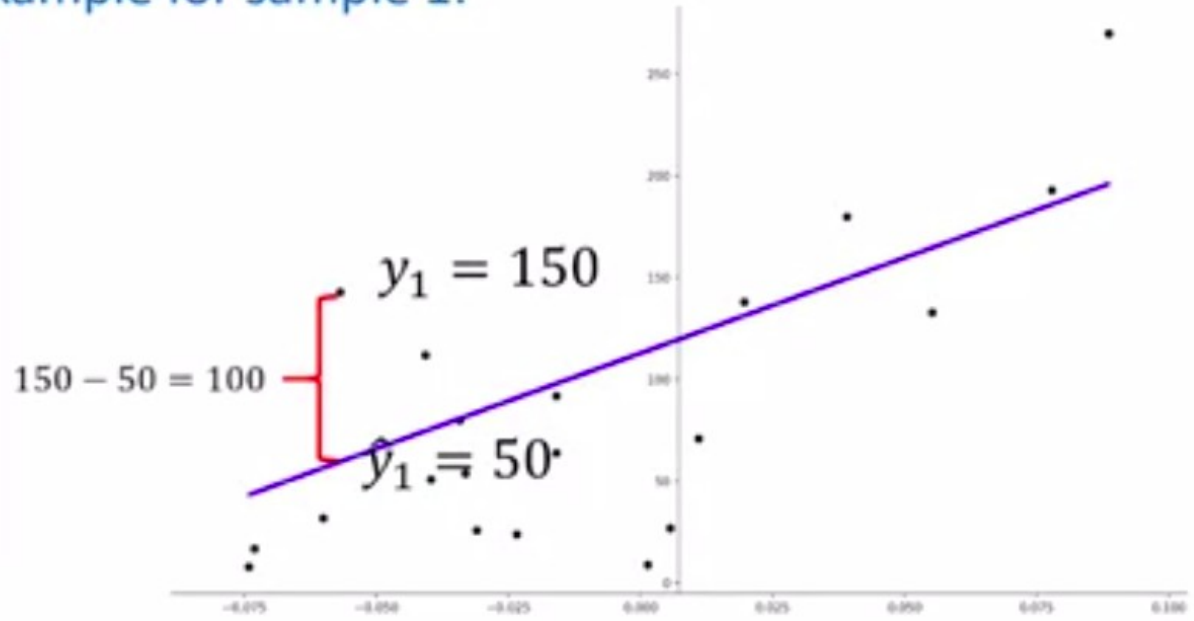
Measures for In-Sample Evaluation

- A way to numerically determine how good the model fits on dataset.

- Two important measures to determine the fit of a model:
  - **Mean Squared Error (MSE)**
  - **R-squared (R^2)**

Mean Squared Error (MSE)

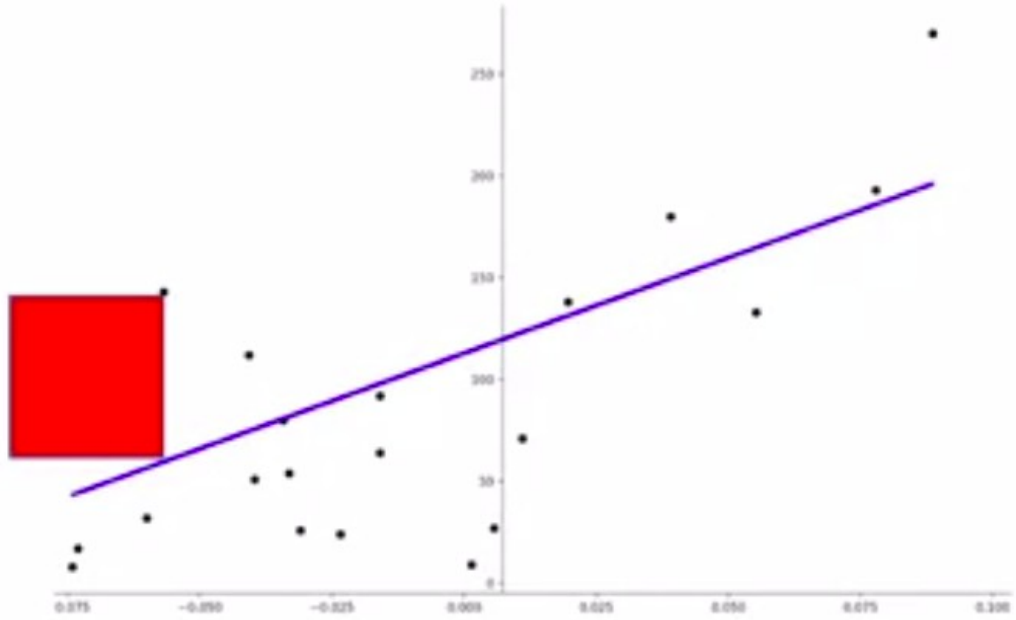
- For Example for sample 1:



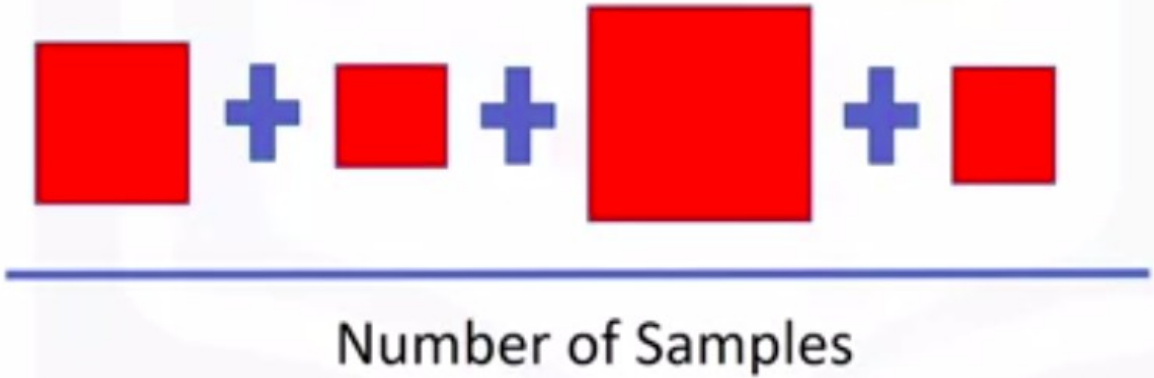
Mean Squared Error (MSE)

- To make all the values positive we square it

$(100)^2$



Mean Squared Error (MSE)



Question

How would you calculate the mean squared error between your predicted values **Yhat** and actual values **Y**?

```
1  from  sklearn.metrixs import mean_squared_error
2
3  mean_squared_error(Y,Yhat)
```

```
1  X = df[['highway-mpg']]
2  Y = df['price']
3  lm.fit(X, Y)
4  lm.score(Yhat,Y)
5
```

Correct

correct

Skip Continue

Mean Squared Error (MSE)

- In python we can measure the MSE as follows

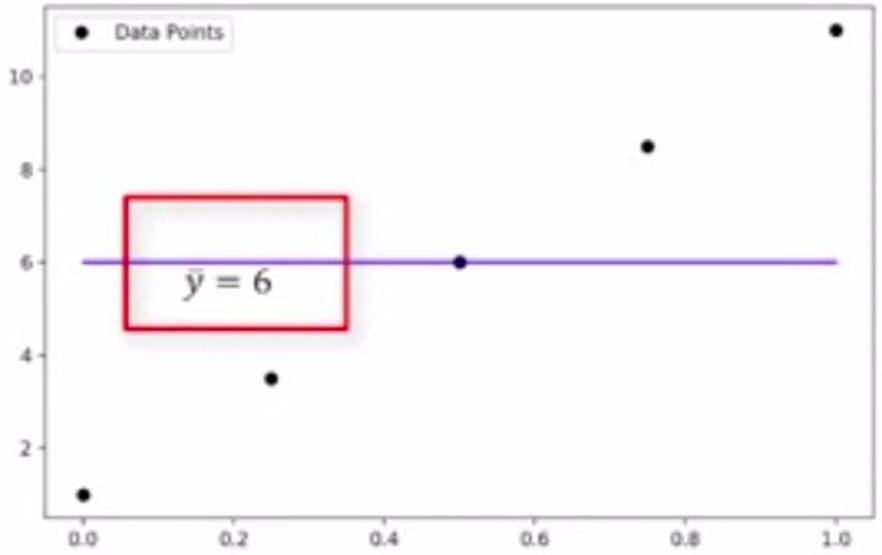
```
from  sklearn.metrics import mean_squared_error

mean_squared_error(df['price'],Y_predict_simple_fit)


3163502.944639888
```

R-squared/ R^2

- The Coefficient of Determination or R squared (R^2)
- Is a measure to determine how close the data is to the fitted regression line.
- R^2: the percentage of variation of the target variable (Y) that is explained by the linear model.
- Think about as comparing a regression model to a simple model i.e the mean of the data points

Coefficient of Determination (R^2)

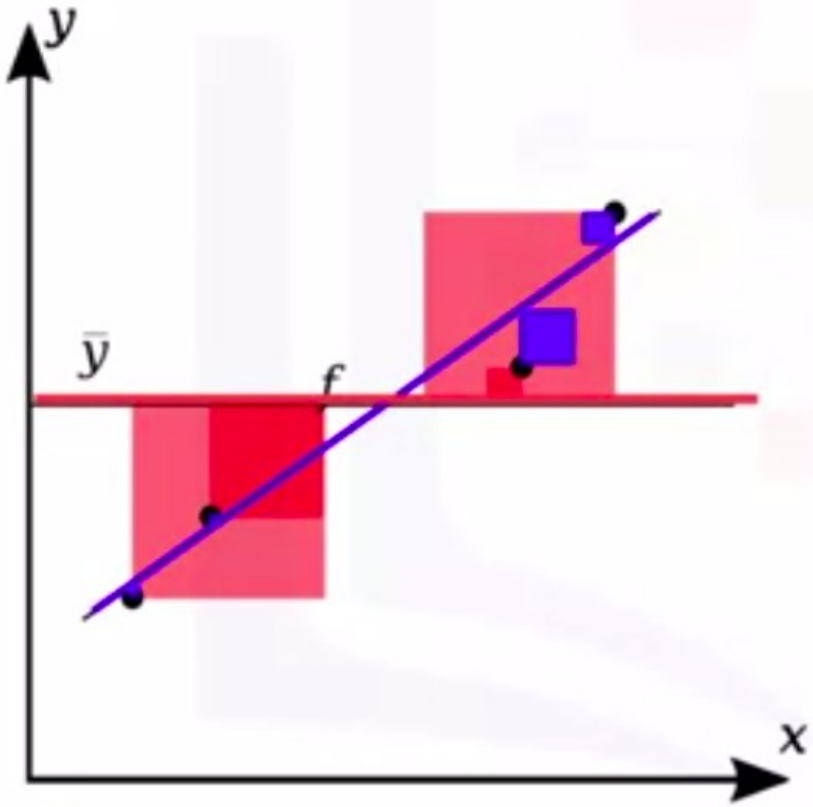
- In this example the average of the data points $\bar{y}$ is 6



Coefficient of Determination (R^2)

$$R^2 = \left(1 - \frac{\text{MSE of regression line}}{\text{MSE of the average of the data}}\right)$$

Coefficient of Determination (R^2)

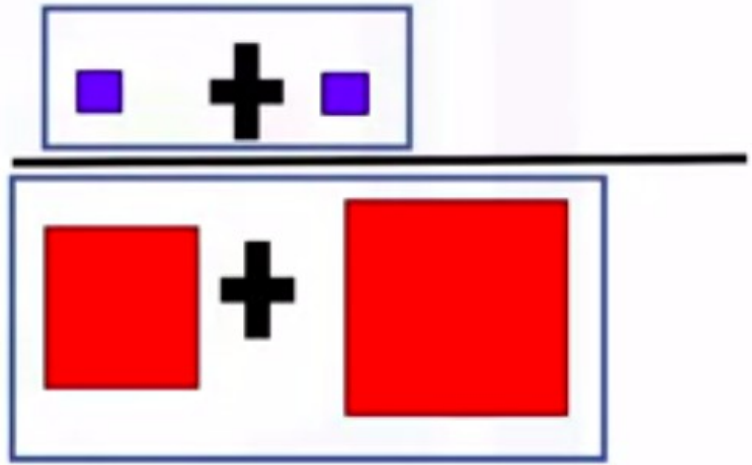


- The blue line represents the regression line
- The blue squares represents the MSE of the regression line
- The red line represents the average value of the data points
- The red squares represent the MSE of the red line
- We see the area of the blue squares is much smaller than the area of the red squares

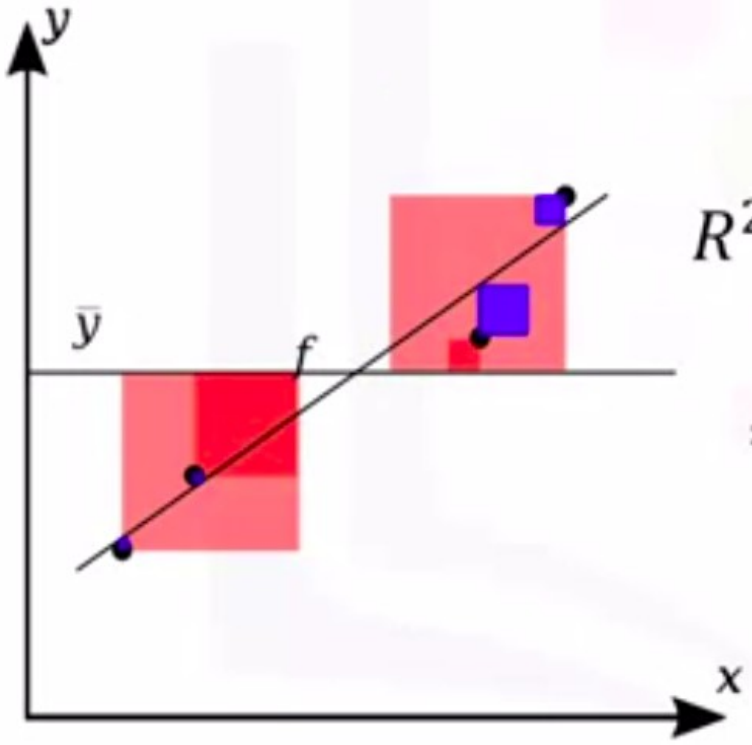
Coefficient of Determination (R^2)

- In this case ratio of the areas of MSE is close to zero

$$\frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}} = 0$$



Coefficient of Determination (R^2)



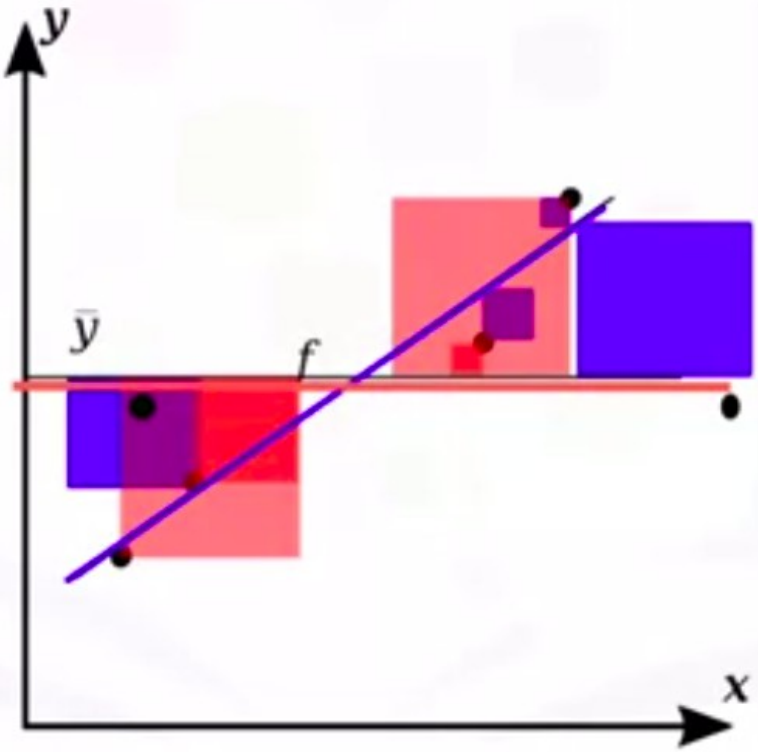
$$R^2 = \left(1 - \frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}}\right)$$

$$= (1 - 0)$$

$$= (1 - 0)$$

$$= 1$$

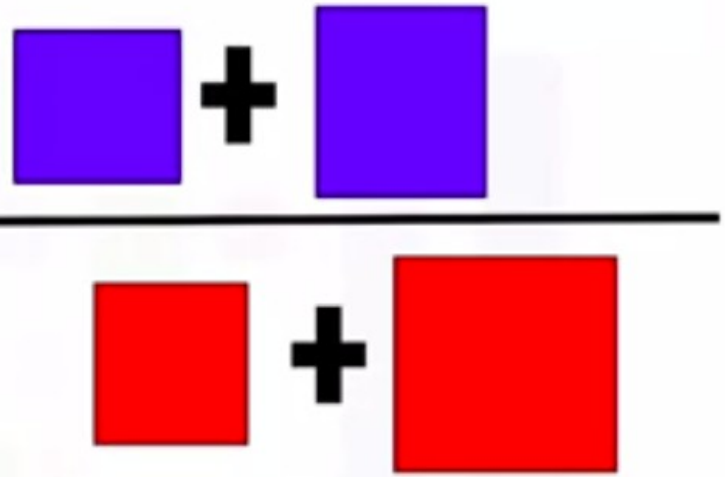
Coefficient of Determination (R^2)



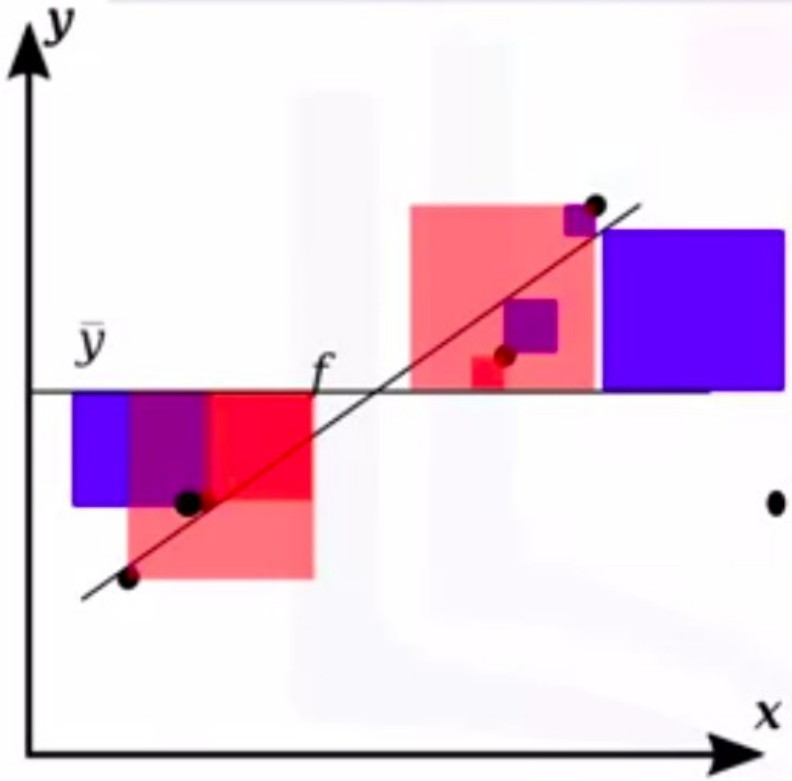
Source: https://en.wikipedia.org/wiki/Coefficient_of_determination

Coefficient of Determination (R^2)

$$\frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}} = \frac{\blacksquare + \blacksquare}{\blacksquare + \blacksquare} = 1$$



Coefficient of Determination (R^2)



$$R^2 = \left(1 - \frac{\text{MSE of regression line}}{\text{MSE of } \bar{y}}\right)$$

$$= (1 - 1)$$

$$= 0$$

R-squared/ R^2

- Generally the values of the MSE are between 0 and 1.
- We can calculate the R^2 as follows

```
X = df[['highway-mpg']]
Y = df['price']

lm.fit(X, Y)
```

```
lm.score(X,y)
0.496591188
```

Question

Consider the following lines of code:

```
X = df[['highway-mpg']]
Y = df['price']
lm.fit(X, Y)
```

How would you calculate the R^2 for **X** and **Y**?

lm.score(X,Y)

Correct

correct

Skip Continue

R-squared/ R^2

- Generally the values of the MSE are between 0 and 1.
- We can calculate the R^2 as follows

```
X = df[['highway-mpg']]
Y = df['price']

lm.fit(X, Y)

lm.score(X,y)
0.496591188
```

Prediction and Decision Making

Decision Making: Determining a Good Model Fit

To determine final best fit, we look at a combination of:

- Do the predicted values make sense
- Visualization
- Numerical measures for evaluation
- Comparing Models

Do the predicted values make sense

- First we train the model

```
lm.fit(df['highway-mpg'],df['prices'])
```

- Let's predict the price of a car with 30 highway-mpg.

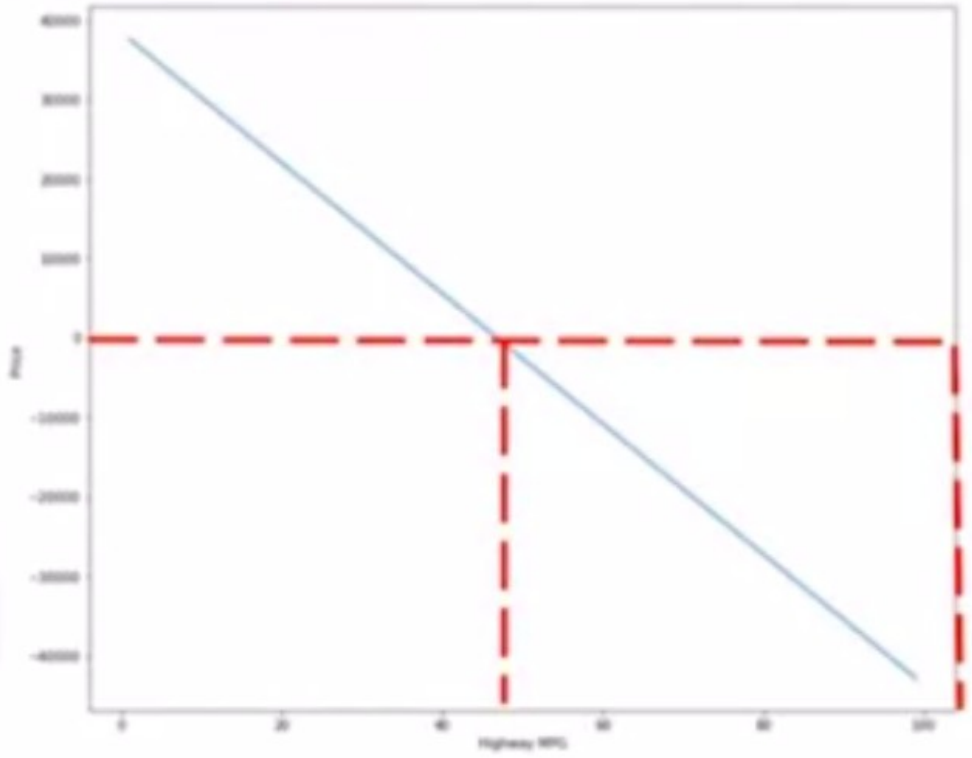
```
lm.predict(np.array(30.0).reshape(-1,1))
```

- Result: $ 13771.30

```
lm.coef_
-821.73337832
```

  - **Price** = 38423.31 - 821.73 * **highway-mpg**

Do the predicted values make sense



Do the predicted values make sense

- First we import numpy

```
import numpy as np
```

- We use the numpy function arrange to generate a sequence from 1 to 100

```
new_input=np.arange(1,101,1).reshape(-1,1)
```

Do the predicted values make sense

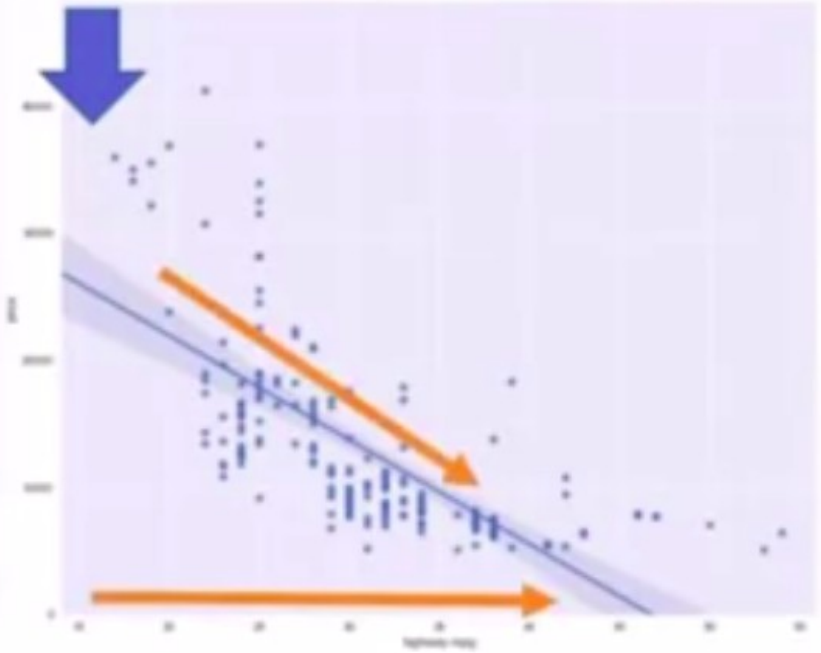
- We can predict new values

```
yhat=lm.predict(new_input)
```

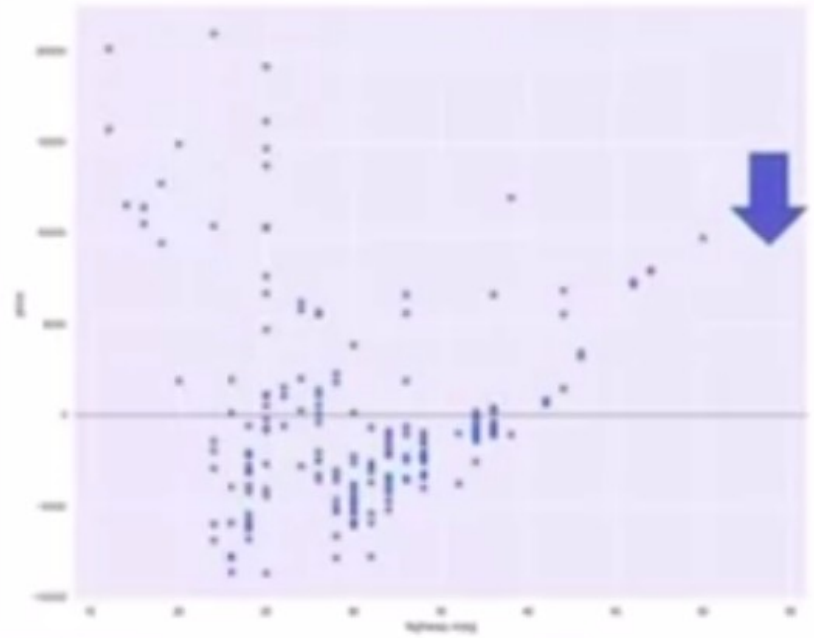
```
array([ 37601.57247984,  36779.83910151,  35958.10572319,  35136.37234487,
        34314.63896655,  33492.90558823,  32671.1722099 ,  31849.43883158,
        31027.70545326,  30205.97207494,  29384.23869662,  28562.50531829,
        27740.77193997,  26919.03856165,  26097.30518333,  25275.57180501,
        24453.83842668,  23632.10504836,  22810.37167004,  21988.63829172,
        21166.9049134 ,  20345.17153508,  19523.43815675,  18701.70477843,
        17879.97140011,  17058.23802179,  16236.50464347,  15414.77126516,
        14593.03788682,  13771.3045085 ,  12949.57113018,  12127.83775186,
        11306.10437353,  10484.37099521,   9662.63761689,   8840.90423857,
         8019.17086025,   7197.43748192,   6375.7041036 ,   5553.97072528,
         4732.23734696,   3910.50396864,   3088.77059031,   2267.03721199,
        -1861.62967962,  -2663.36305794,  -3485.09643626,  -4306.82981458,
        -5128.5631929 ,  -5950.29657123,  -6772.02994955,  -7593.76332787,
        -8415.49670619,  -9237.23008451, -10058.96346284, -10880.69684116,
       -11702.43021948, -12524.1635978 , -13345.89697612, -14167.63035445,
       -14989.36373277, -15811.09711109, -16632.8304894 , -17454.56386773,
       -18276.29724604, -19098.03062438, -19919.7640027 , -20741.49738102,
       -21563.23075934, -22384.96413767, -23206.69751599, -24028.43089431,
       -24850.16427263, -25671.89765095, -26493.63102927, -27315.3644076 ,
       -28137.09778592, -28958.83116424, -29780.56454256, -30602.29792088,
       -31424.03129921, -32245.76467753, -33067.49805585, -33889.23143417,
       -34710.96481249, -35532.69819082, -36354.43156914, -37176.16494746,
       -37997.89832578, -38819.6317041 , -39641.36508243, -40463.09846075,
       -41284.83183907, -42106.56521739, -42928.29859571])
```

Visualization

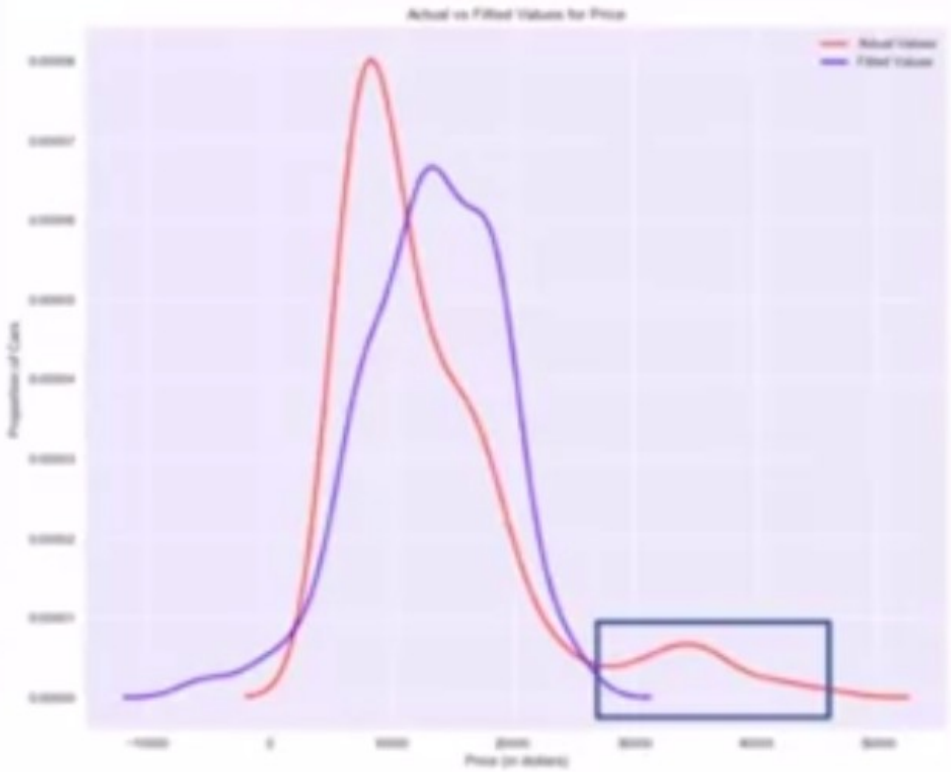
- Simply visualizing your data with a regression



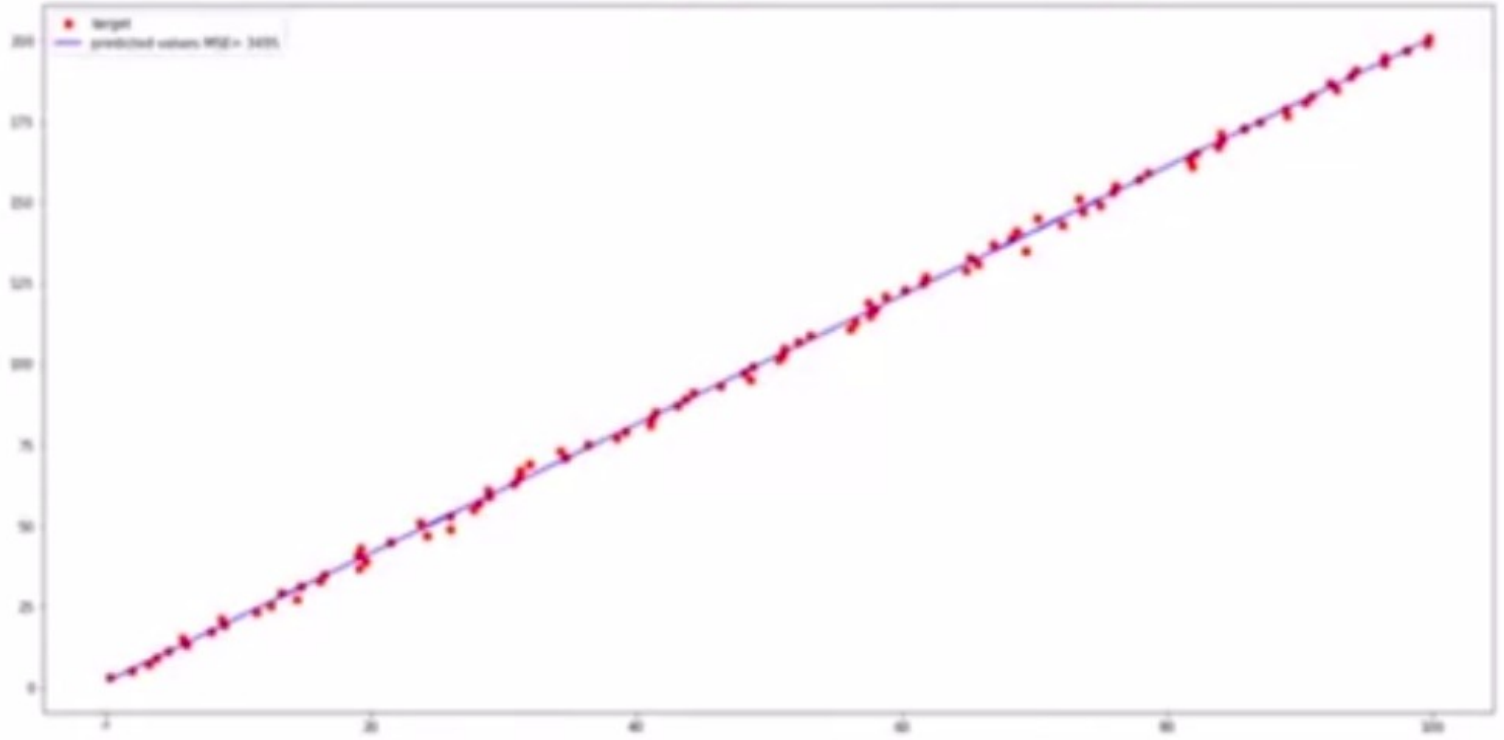
Residual Plot



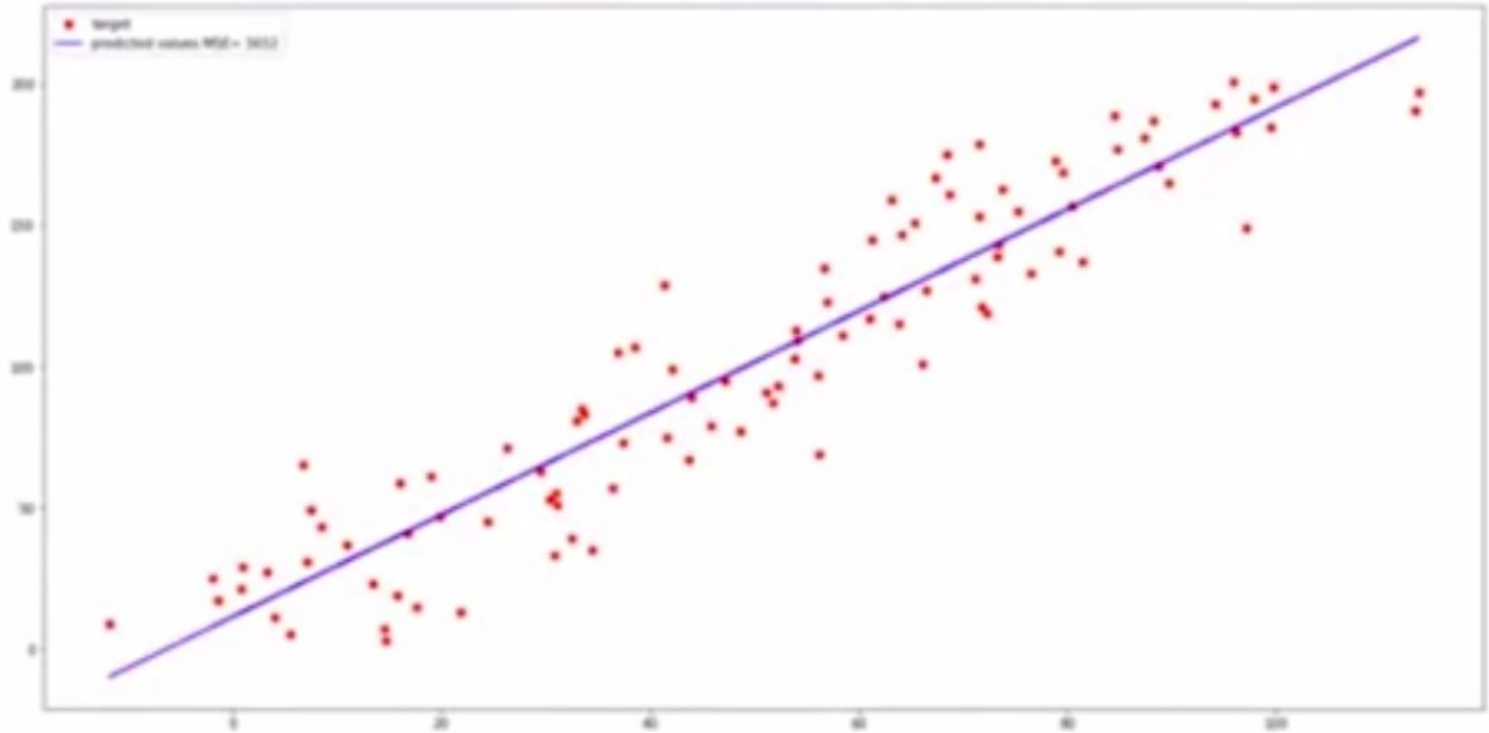
Visualization



Numerical measures for Evaluation



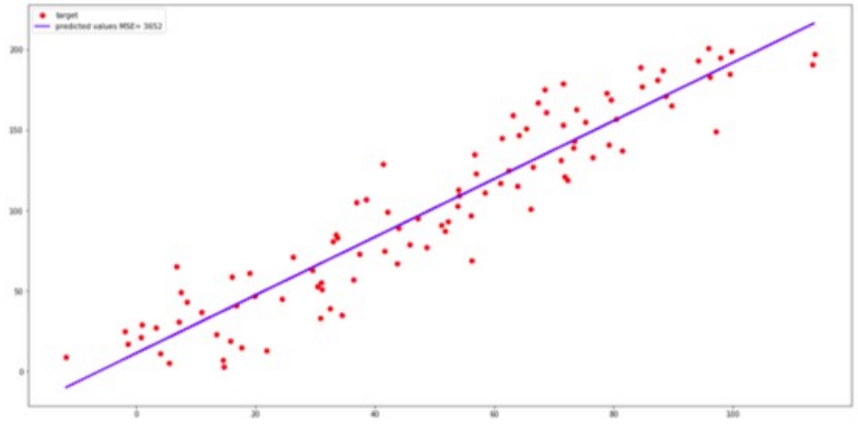
Numerical measures for Evaluation



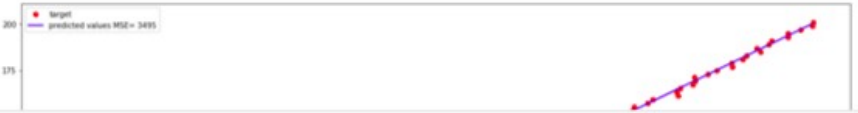
Question

Consider the plots a and b below; red is the sample data and the blue line is the predicted value. Which plot has a higher means square error?

a

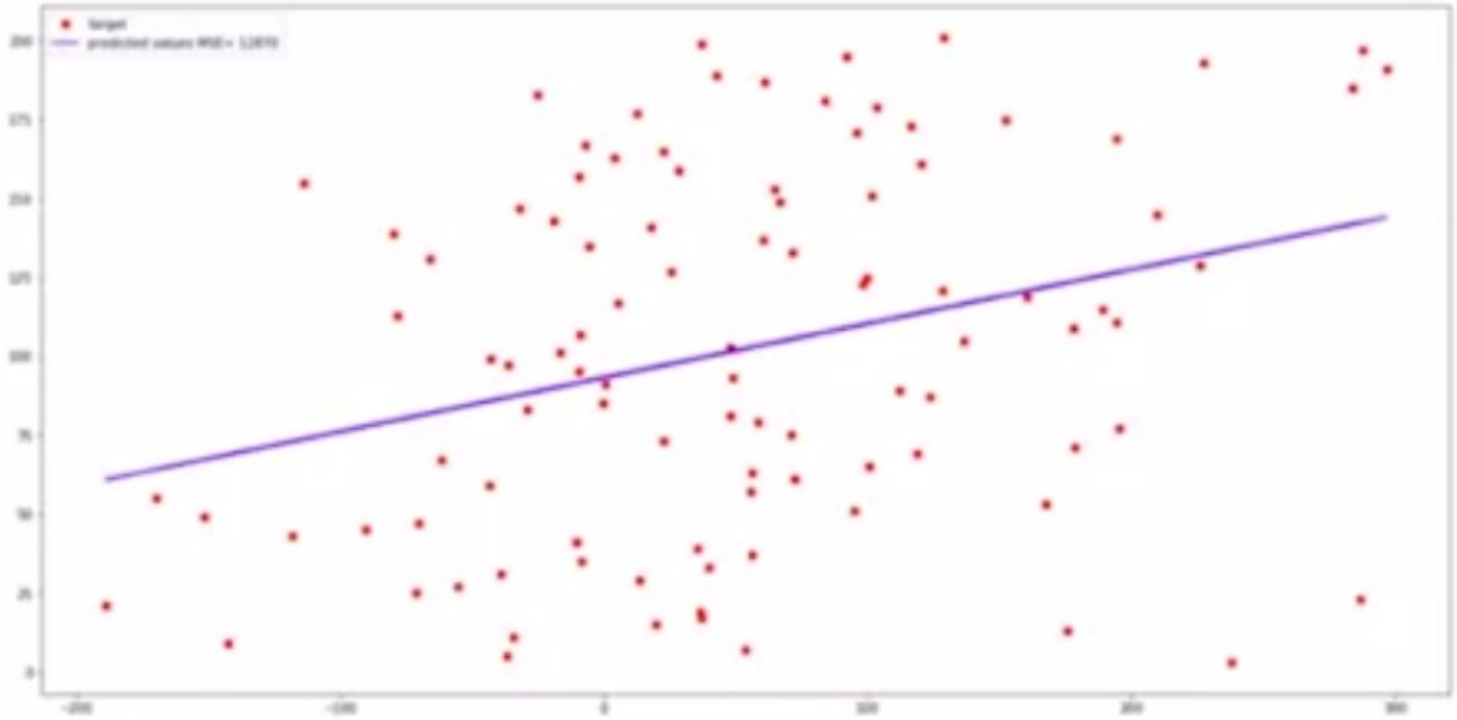


b



Skip Submit

Numerical measures for Evaluation



Lesson Summary

In this lesson, you have learned how to:

Define the explanatory variable and the response variable: Define the response variable (y) as the focus of the experiment and the explanatory variable (x) as a variable used to explain the change of the response variable. Understand the differences between Simple Linear Regression because it concerns the study of only one explanatory variable and Multiple Linear Regression because it concerns the study of two or more explanatory variables.

Evaluate the model using Visualization: By visually representing the errors of a variable using scatterplots and interpreting the results of the model.

Identify alternative regression approaches: Use a Polynomial Regression when the Linear regression does not capture the curvilinear relationship between variables and how to pick the optimal order to use in a model.

Interpret the R-square and the Mean Square Error: Interpret R-square (x 100) as the percentage of the variation in the response variable y that is explained by the variation in explanatory variable(s) x. The Mean Squared Error tells you how close a regression line is to a set of points. It does this by taking the average distances from the actual points to the predicted points and squaring them.